Sample WebLink PNG Animation and Modulation Scripts

Always backup your WebLink project before working with these sample scripts

You can either import the example page into an existing project or create a new, empty project in which to experiment

In this exercise we will import sample scripts for PNG animation and modulation into the Global Functions tab. Changes to global functions should be done very carefully, even small errors in this area can be very difficult to debug.

You can alternately copy/paste these example scripts into the Local Functions tab of each web page where animation or modulation of PNG graphics is taking place

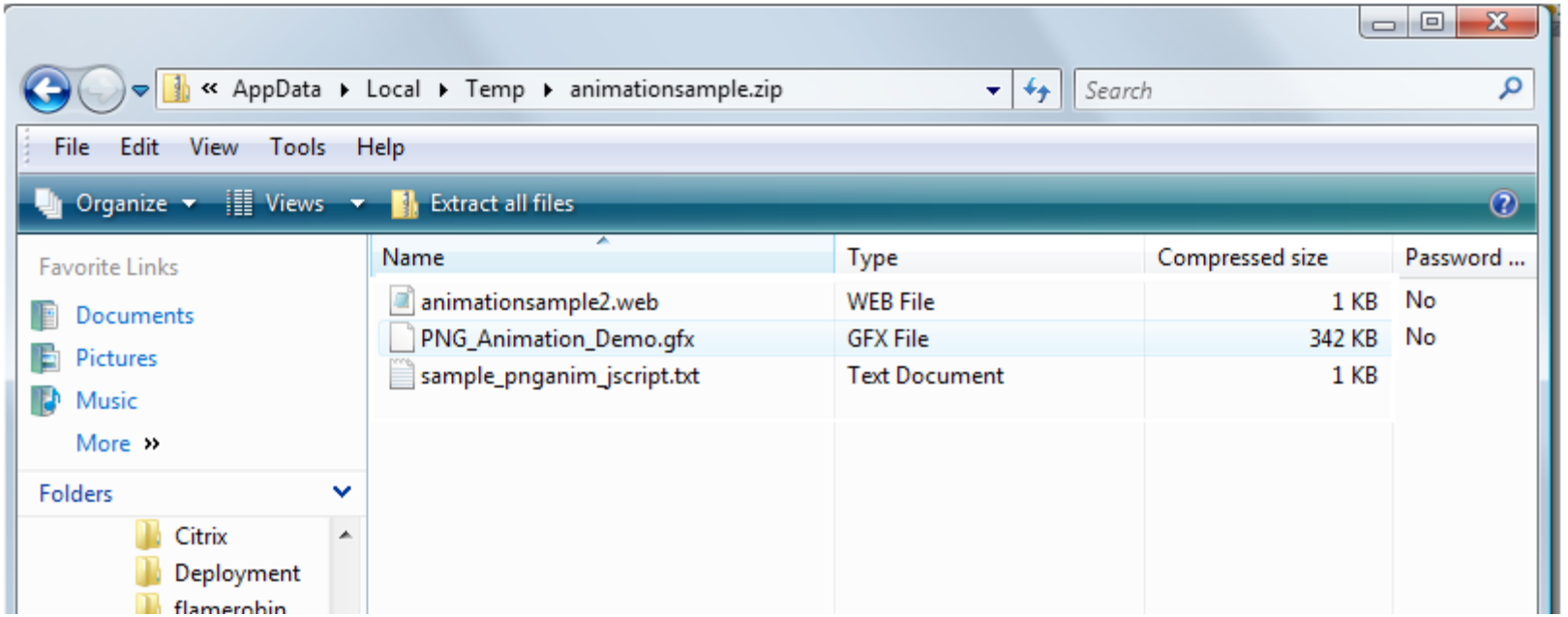
The advantage of importing the sample scripts into Global Functions is you only do this once and the sample scripts are available on any existing or new web page in the project

First we will download a working example of PNG animation and run it in your WebLink to make sure everything we need is in place

After that we will create a new web page with a 2-state PNG animation showing a Fan that spins, depending on the value of a Binary output on an ASIC/2 controller

The we will create an image of a Damper modulating, and finally we will make a Cooling Coil graphic modulate

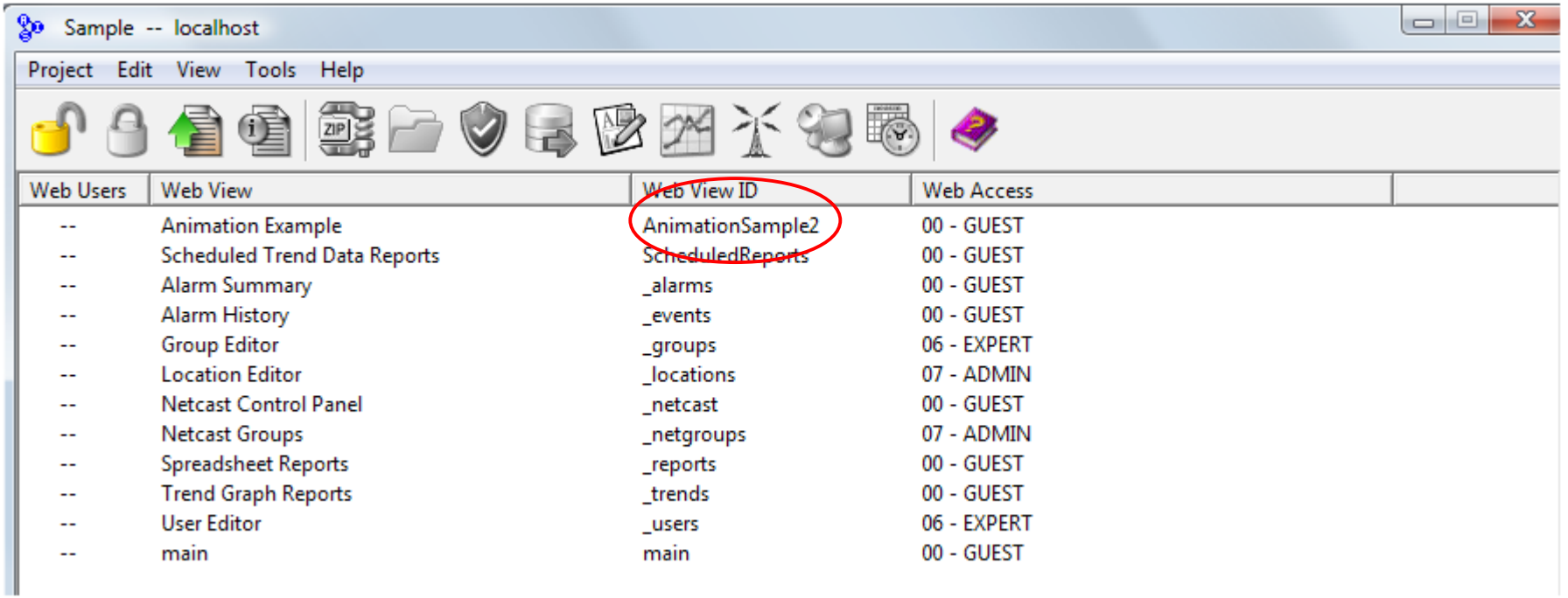
If you you run into any trouble with the examples or need additional examples please email support@asicontrols.com or contact us at 925 866 8808, option 3



Import 2 files you will need directly from the hyperlinks shown down below

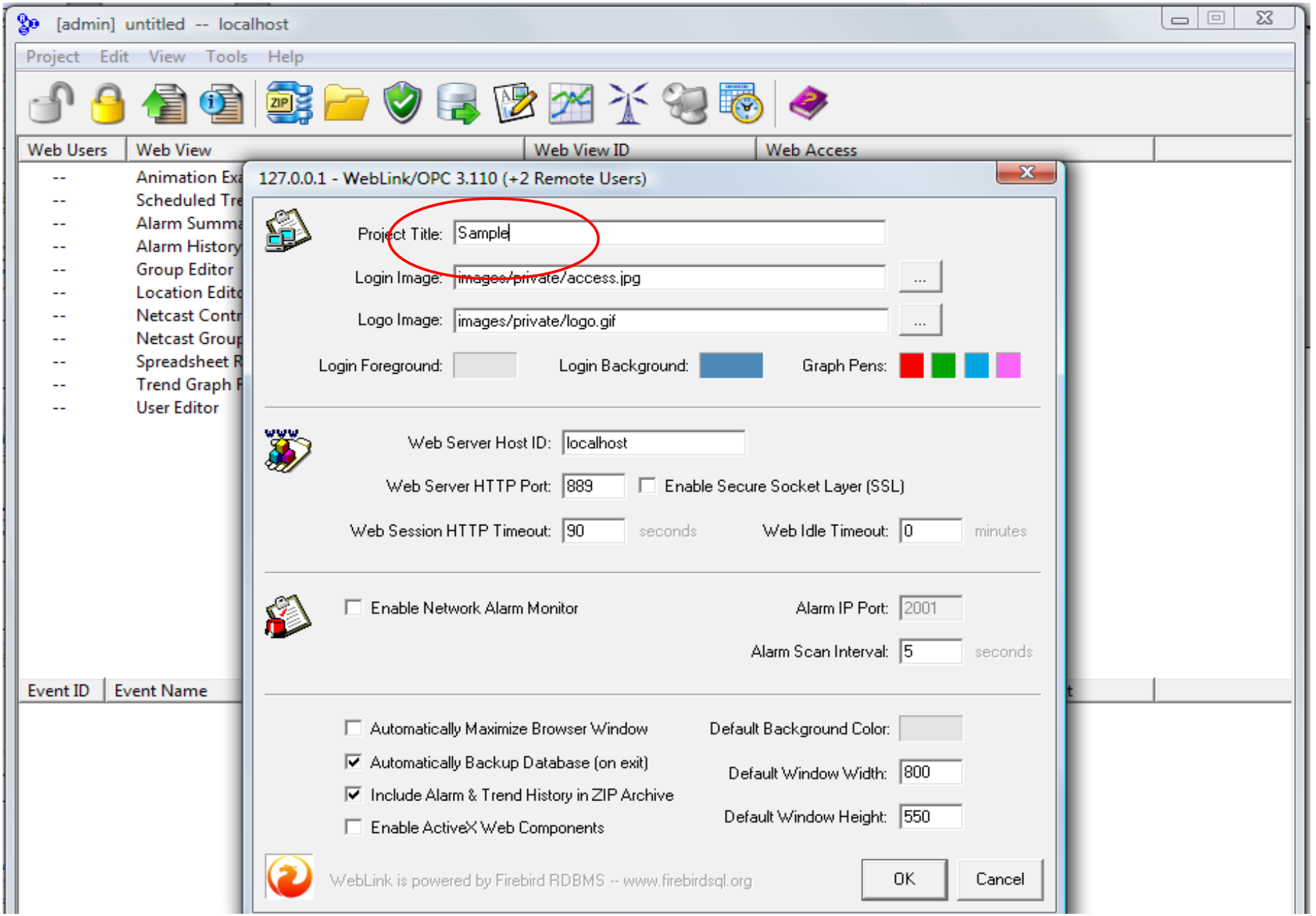
Put the WEB file in "c:\program files\asi controls\weblink\system.3\" folder

The GFX file goes in "c:\program files\asi controls\weblink\system.3\HTML\images\"



Start WebLink

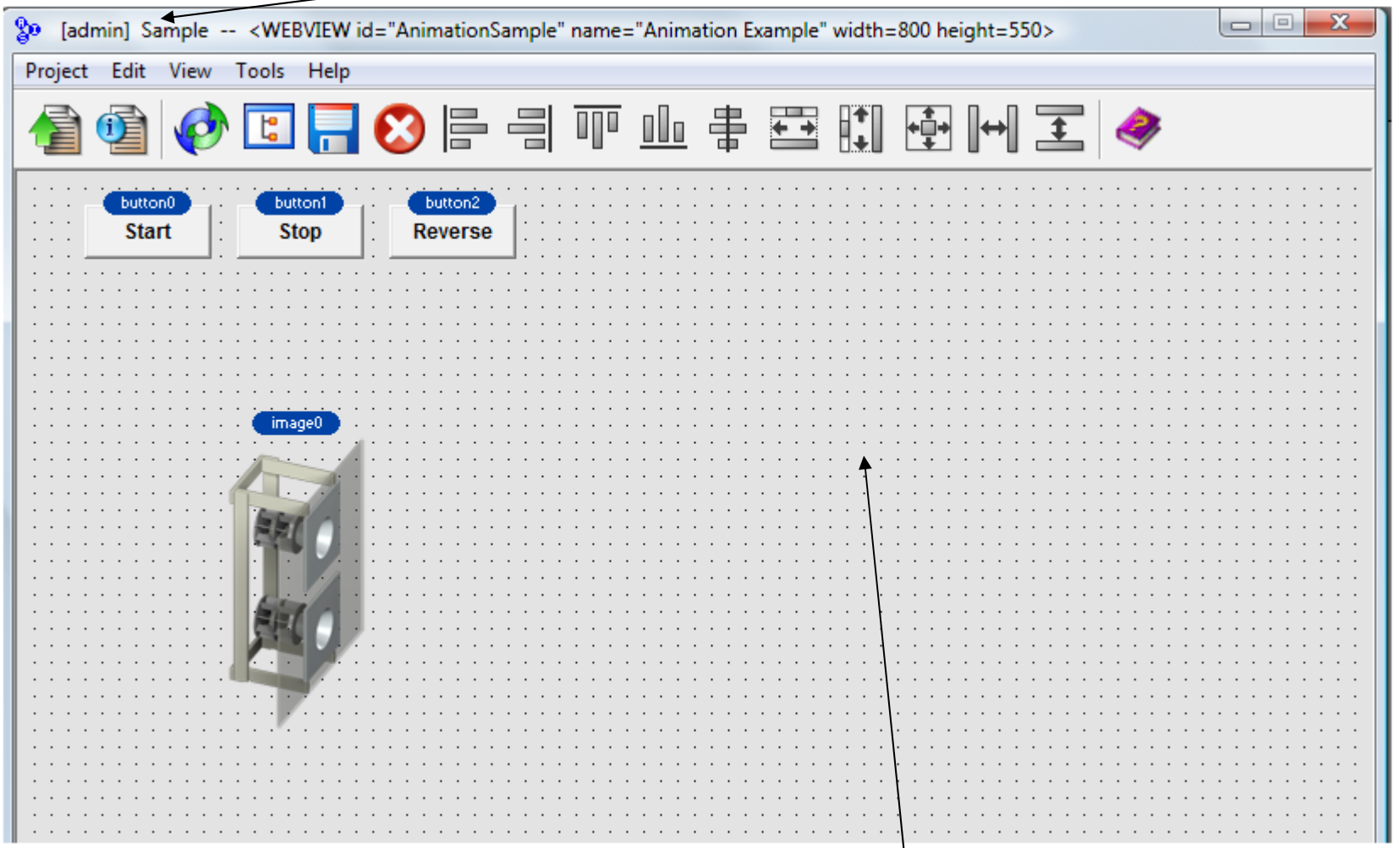
If animationsample2.web was copied to “c:\program files\asi controls\weblink\system.3\” folder then you should see the new screen with Web View ID = AnimationSample2



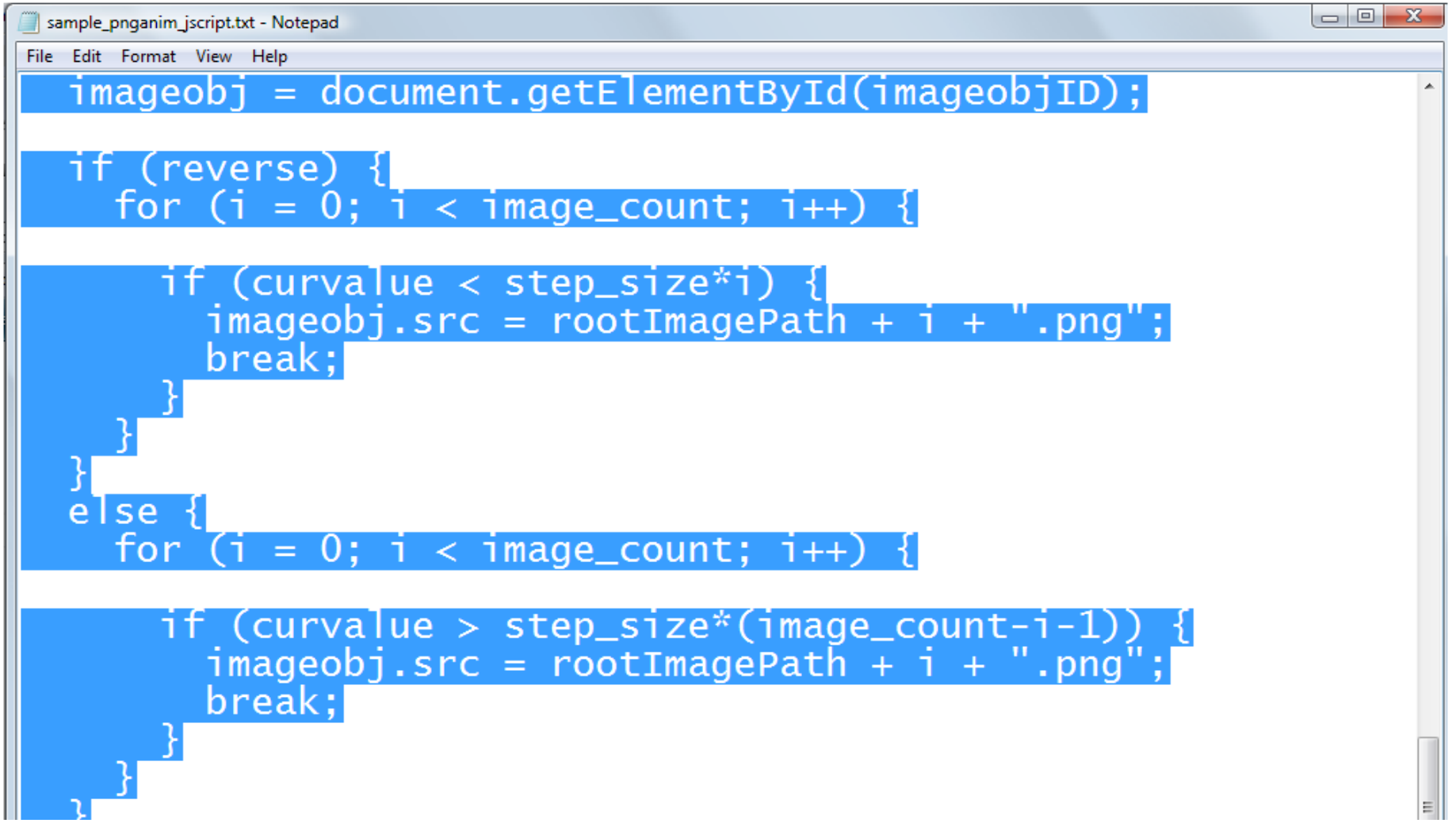
Step 4

You must change **Project Title** to “Sample” in order to work with the Sample GFX graphics

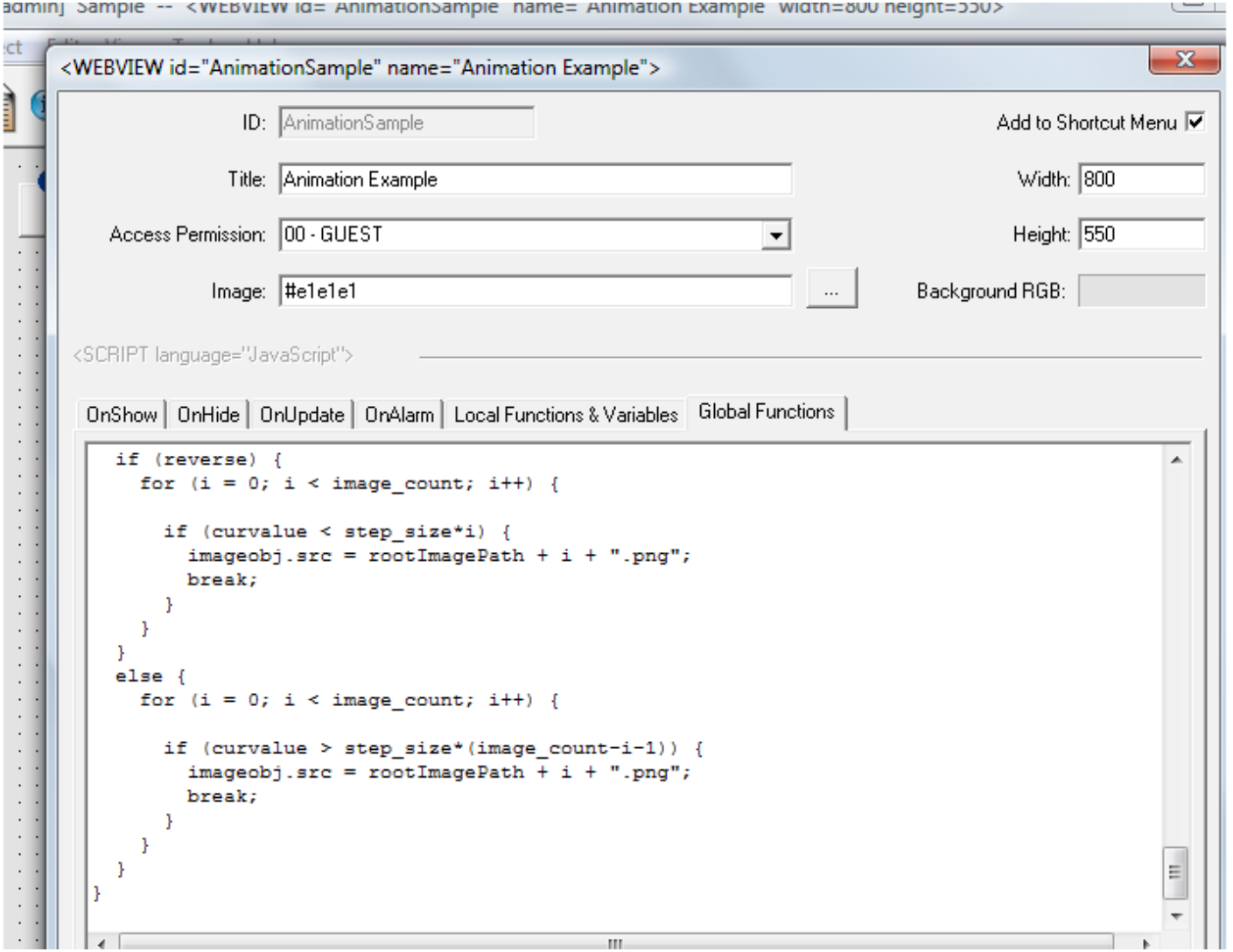
Open the new AnimationSample2 screen. You should see a fan graphic. If you see a blank image the project title is probably wrong. Note the S in Sample must be upper case



Open the Properties for the page by double-clicking somewhere in the grey background area, or by right-clicking and selected Edit Properties.



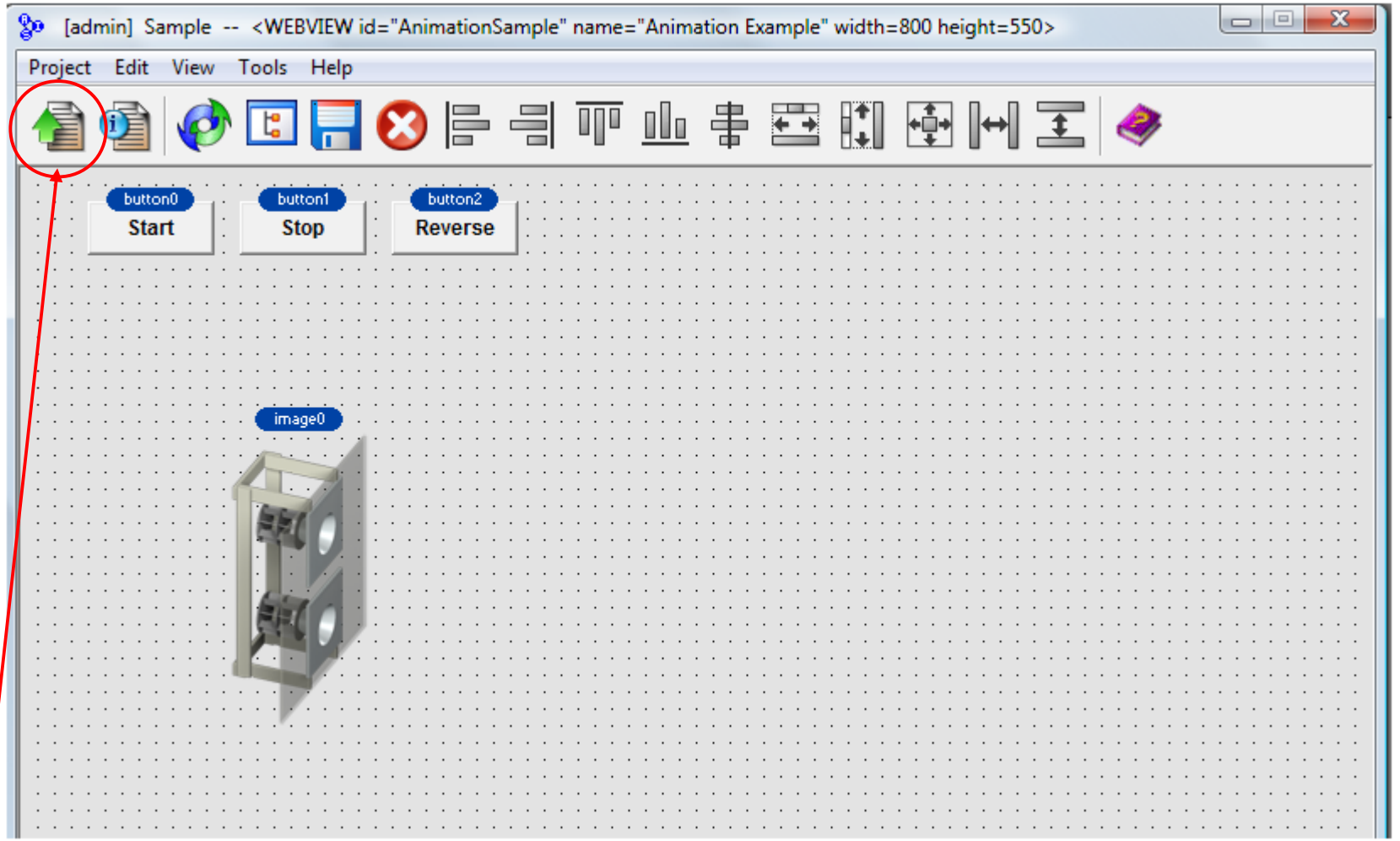
Copy the sample scripts directly from the Notes area below



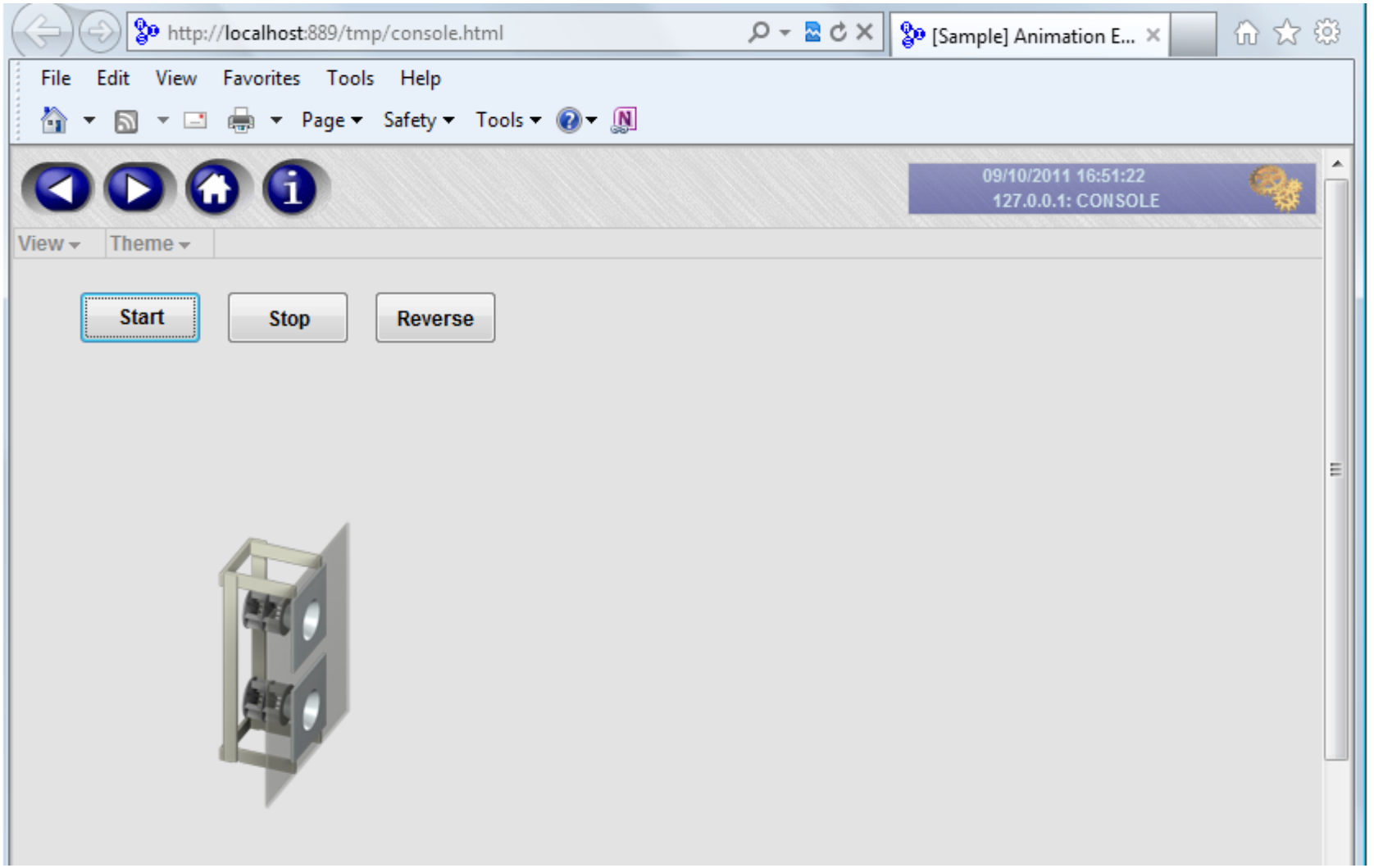
In WebLink, open the screen Properties and go to either the LOCAL or GLOBAL Functions tab. If the tab already contains script scroll down and place the cursor below all existing text

Enter Control-V to paste the text from Windows clipboard into the WebLink editor

You should see the tail end of the pasted text, like above. Click OK to save changes



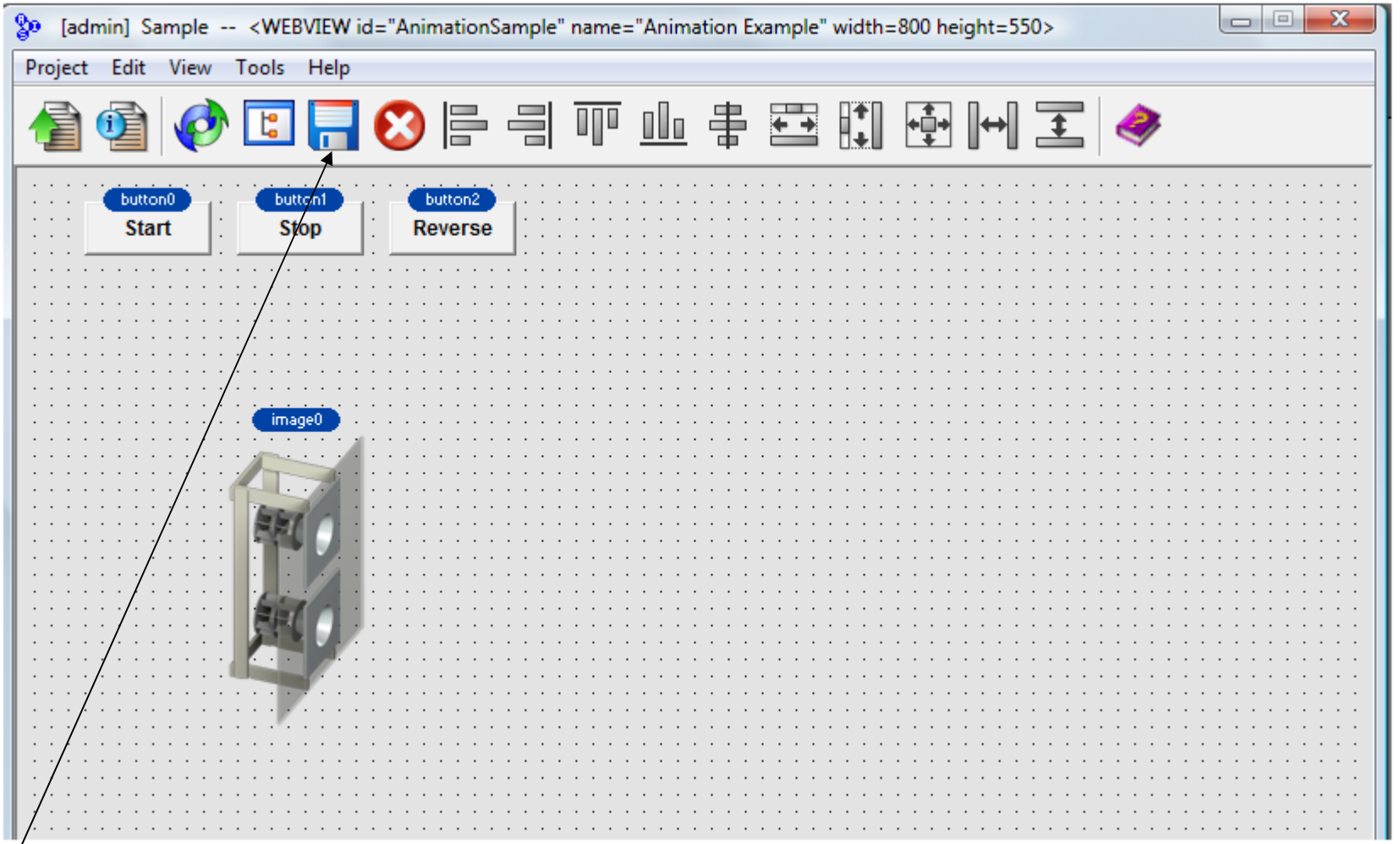
Click the green arrow to start a web session



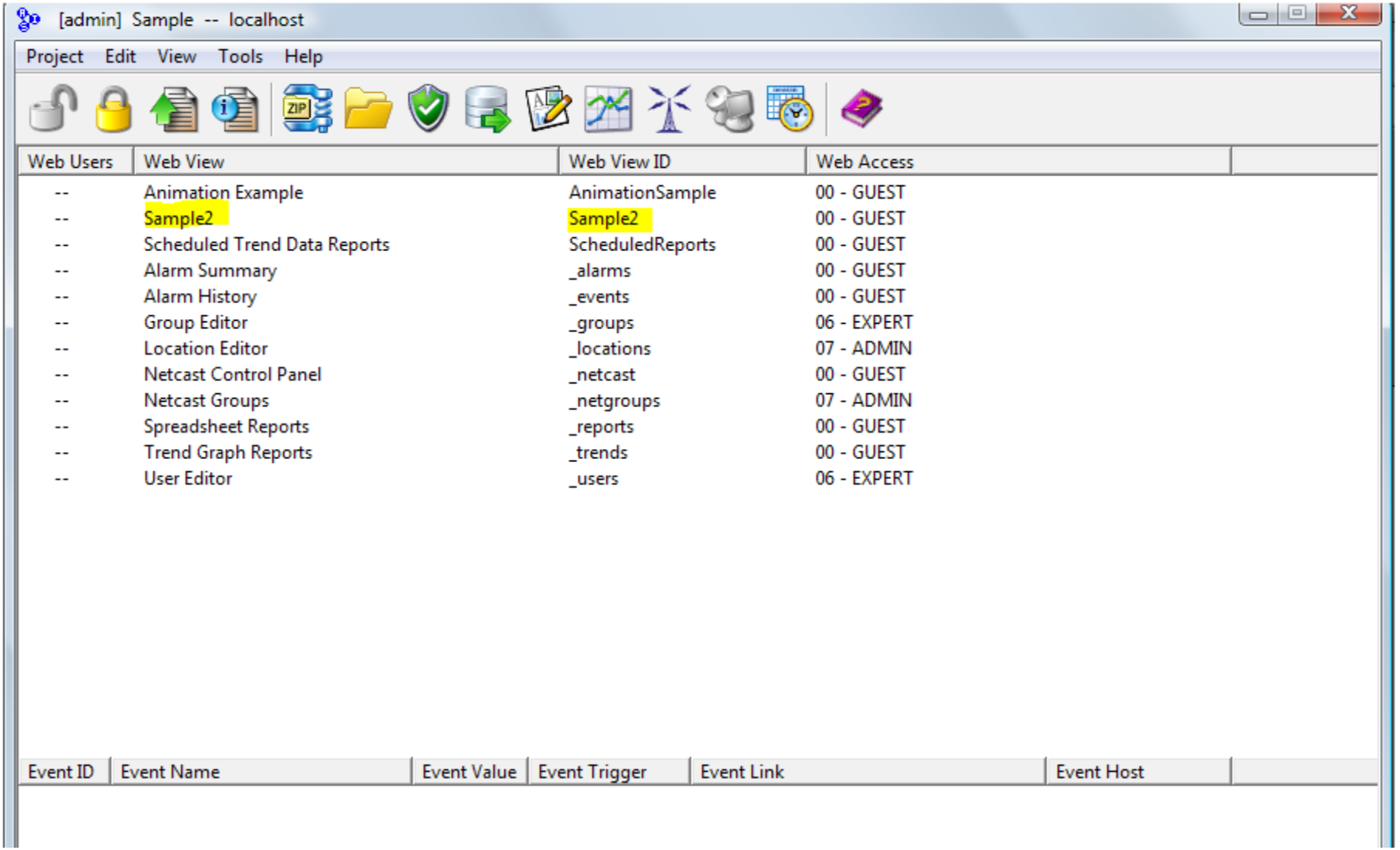
Click the Start button to see the fan rotating, click Reverse to see the fan reverse direction

This exercise demonstrates the basic elements are in place for PNG animation before we attempt to create animations from scratch

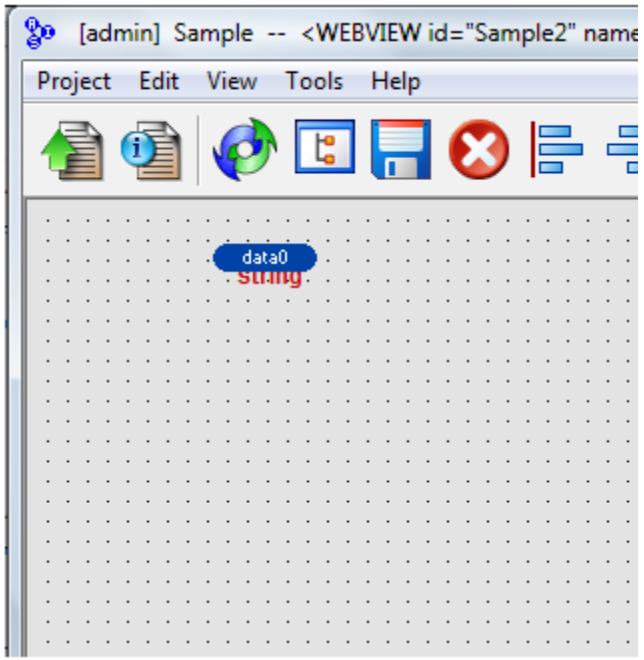
Next we add a screen to do a new fan animation, and show a modulating damper and coil

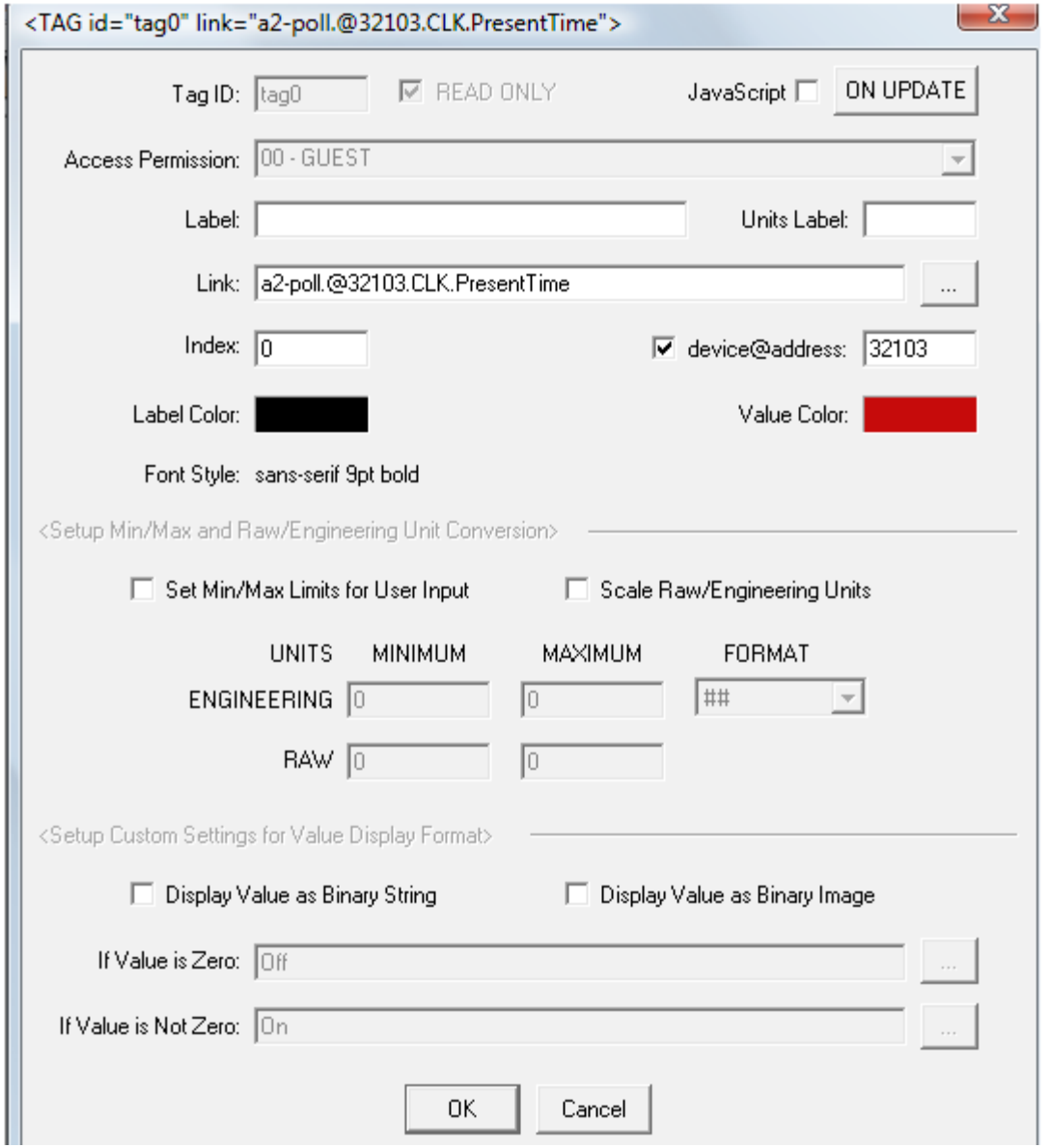


Save the Sample screen and return to the main WebLink console



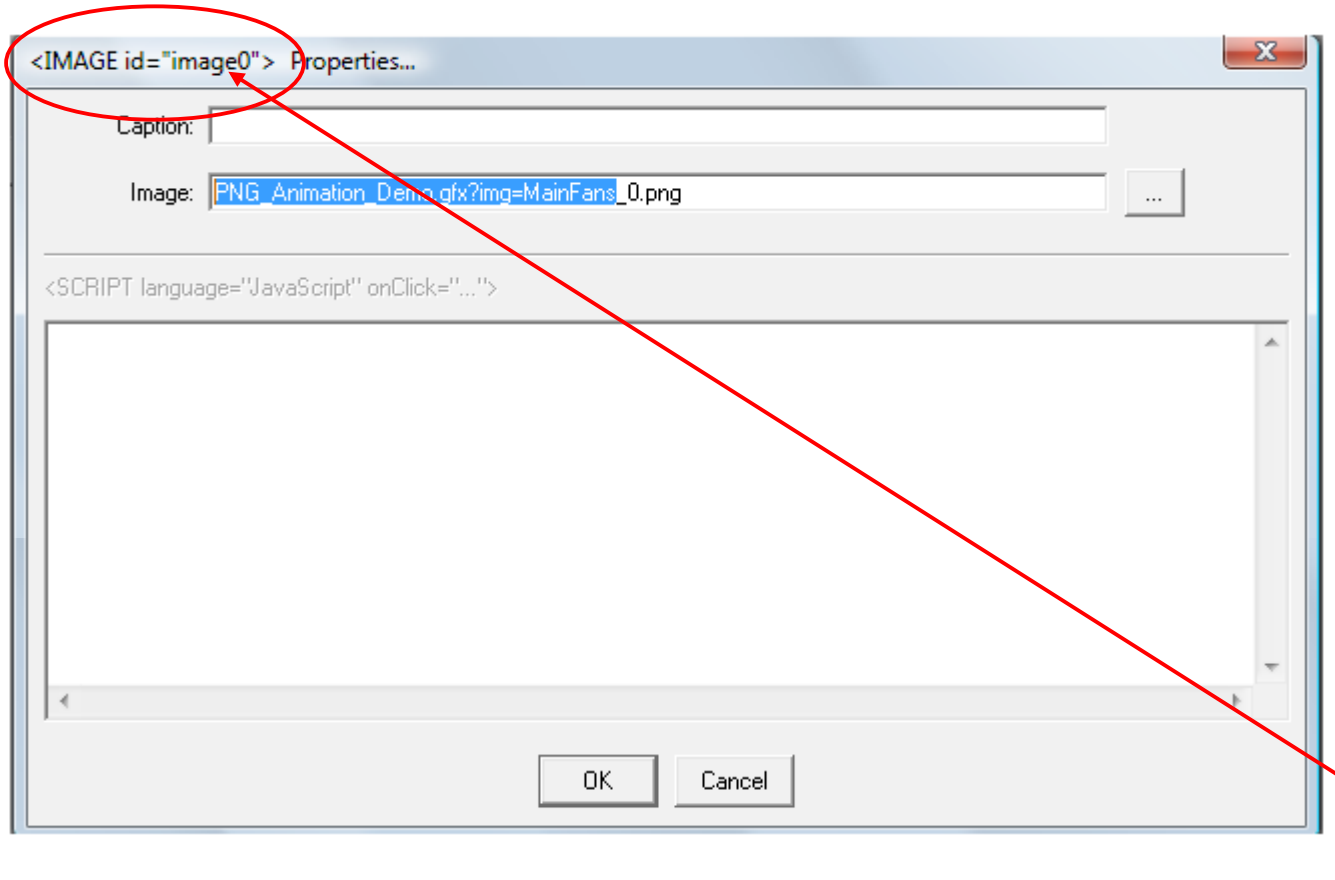
Create a new web screen (web view) named Sample2 and open it for editing





Open the OPC tag browser and drag a tag for an ASIC/2 clock (CLK) object Present TIme value onto the screen. The element displaying the time will have ID of “tag0”

This tag is a convenience item, the time ticking over verifies that communications are OK

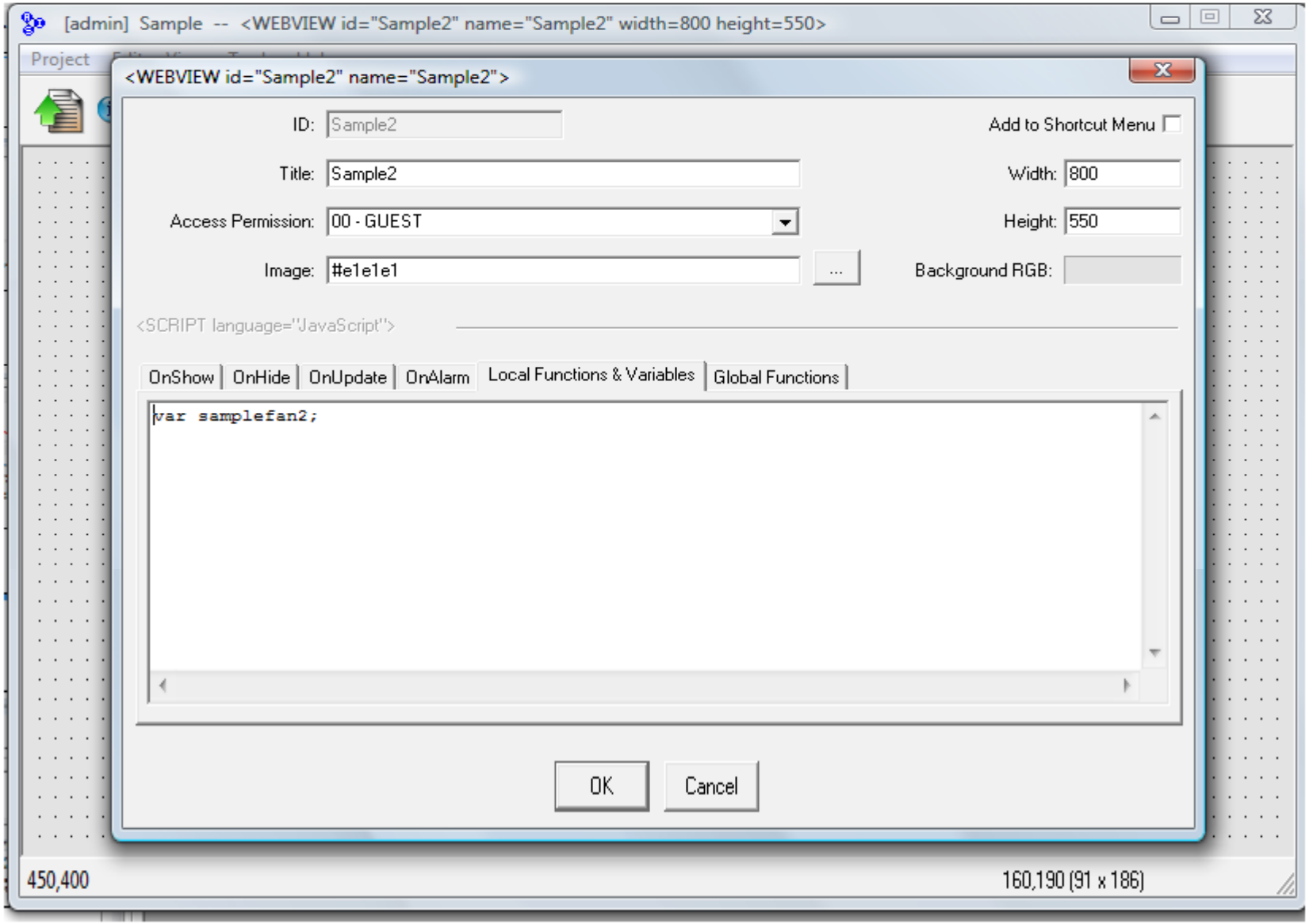


Next, insert an image onto the web screen. *Note down that this new image is "image0"*

Open PNG_Animation_Demo.GFX that you downloaded earlier, select "MainFans_0.png" from the pop-up list. *Note down that there are 3 images total for this fan*

TIP: highlight from start of image name to just before the under-bar ("_") near the end of the line, not including under-bar. Hit Control-C to copy highlighted text to clipboard for later

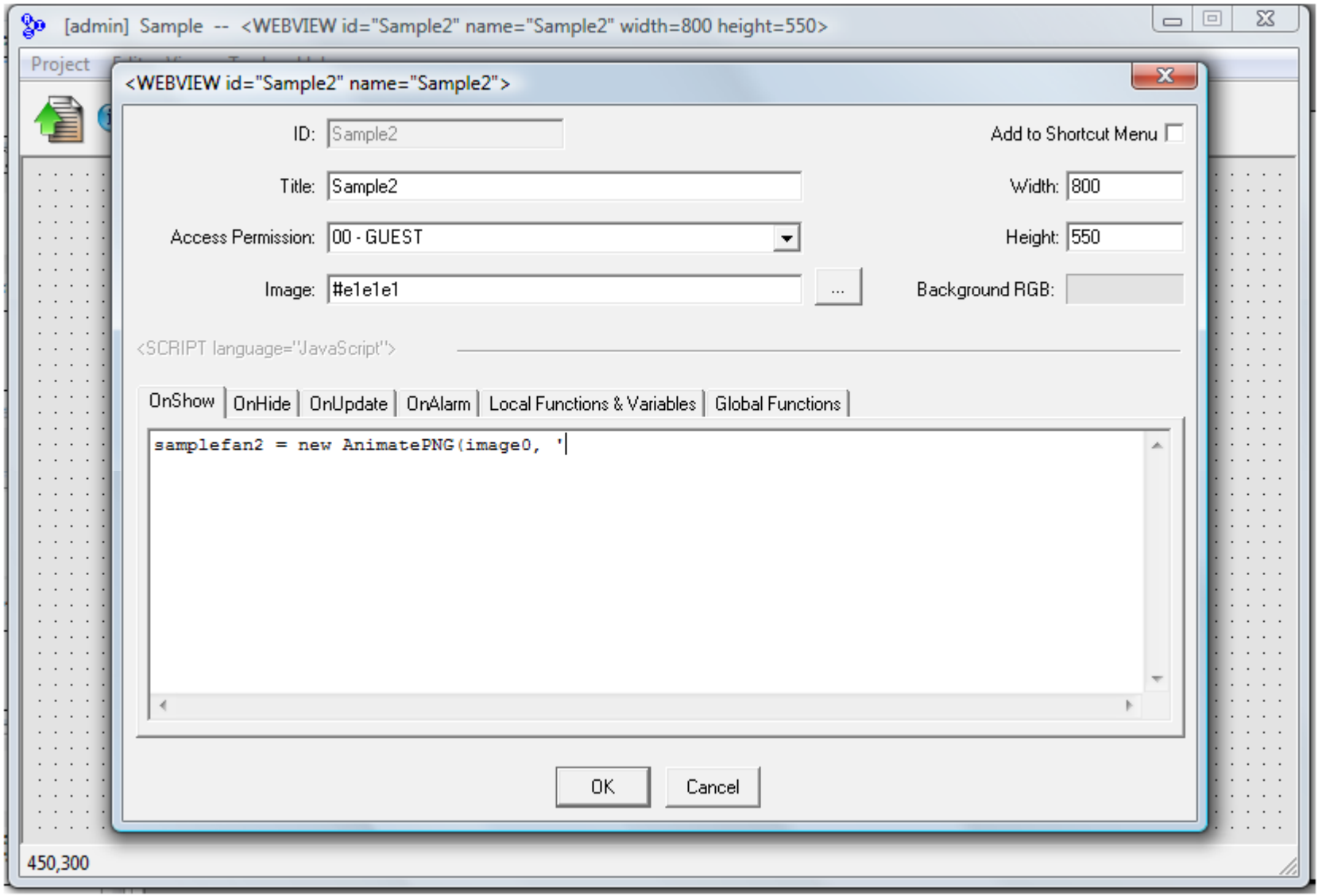
Click OK to save the changes



Create a variable “samplefan2” on the LOCAL FUNCTIONS tab by typing:

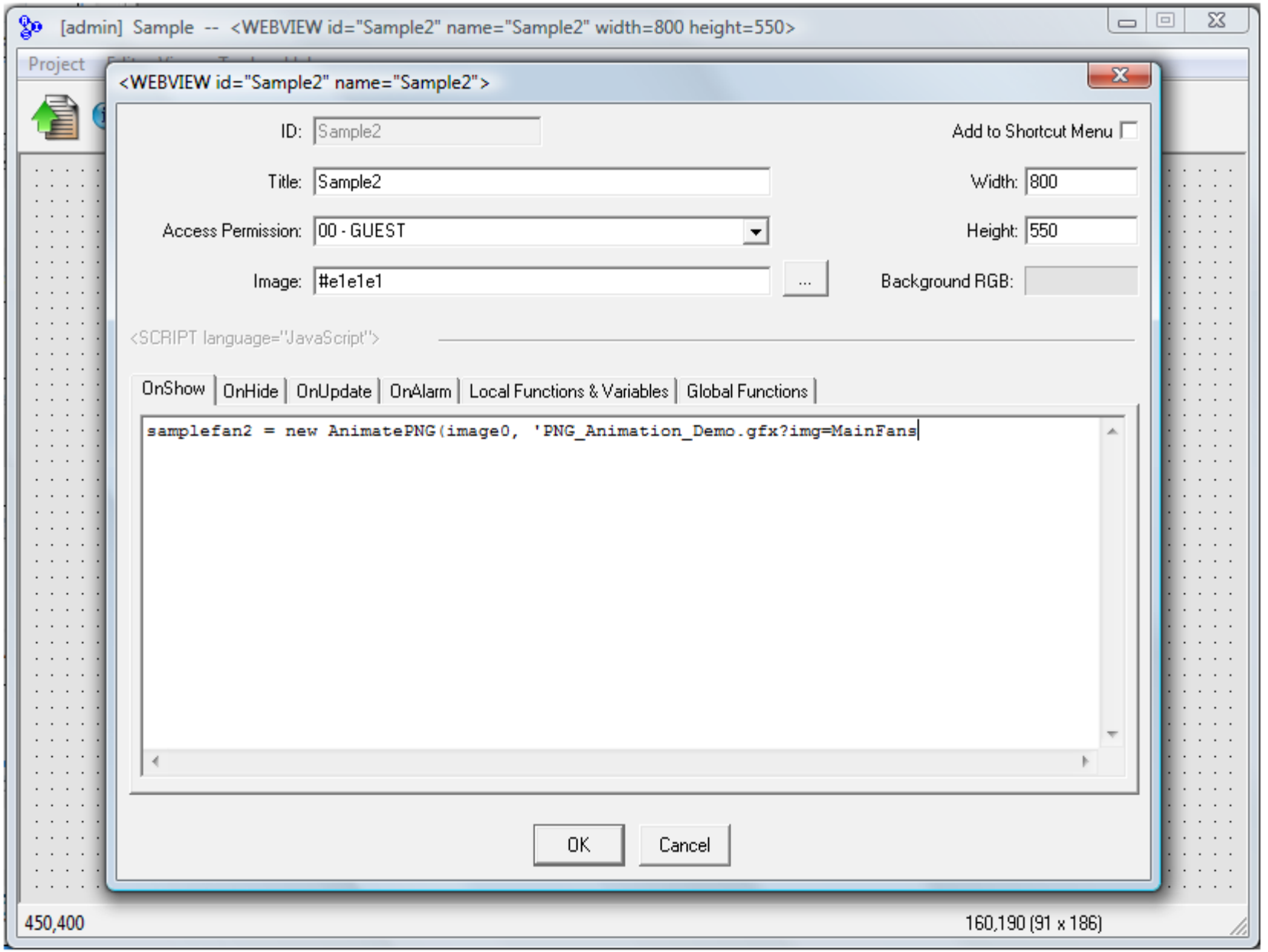
```
var samplefan2;
```

Go to the ON SHOW tab

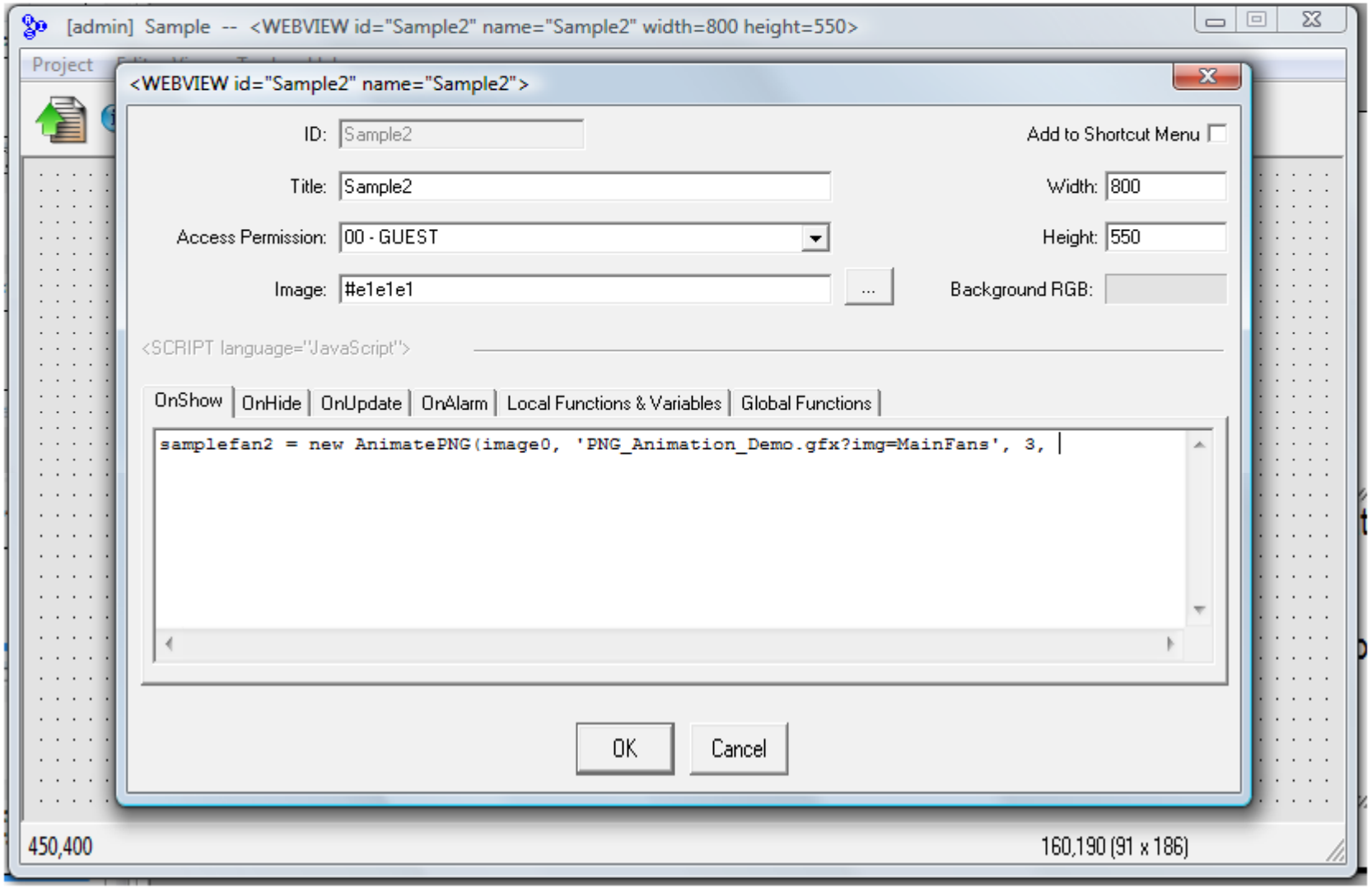


Enter the following text. Remember that earlier we note “image0” as the name of the new image we are going to animate:

```
samplefan2 = new AnimatePNG(image0, '
```

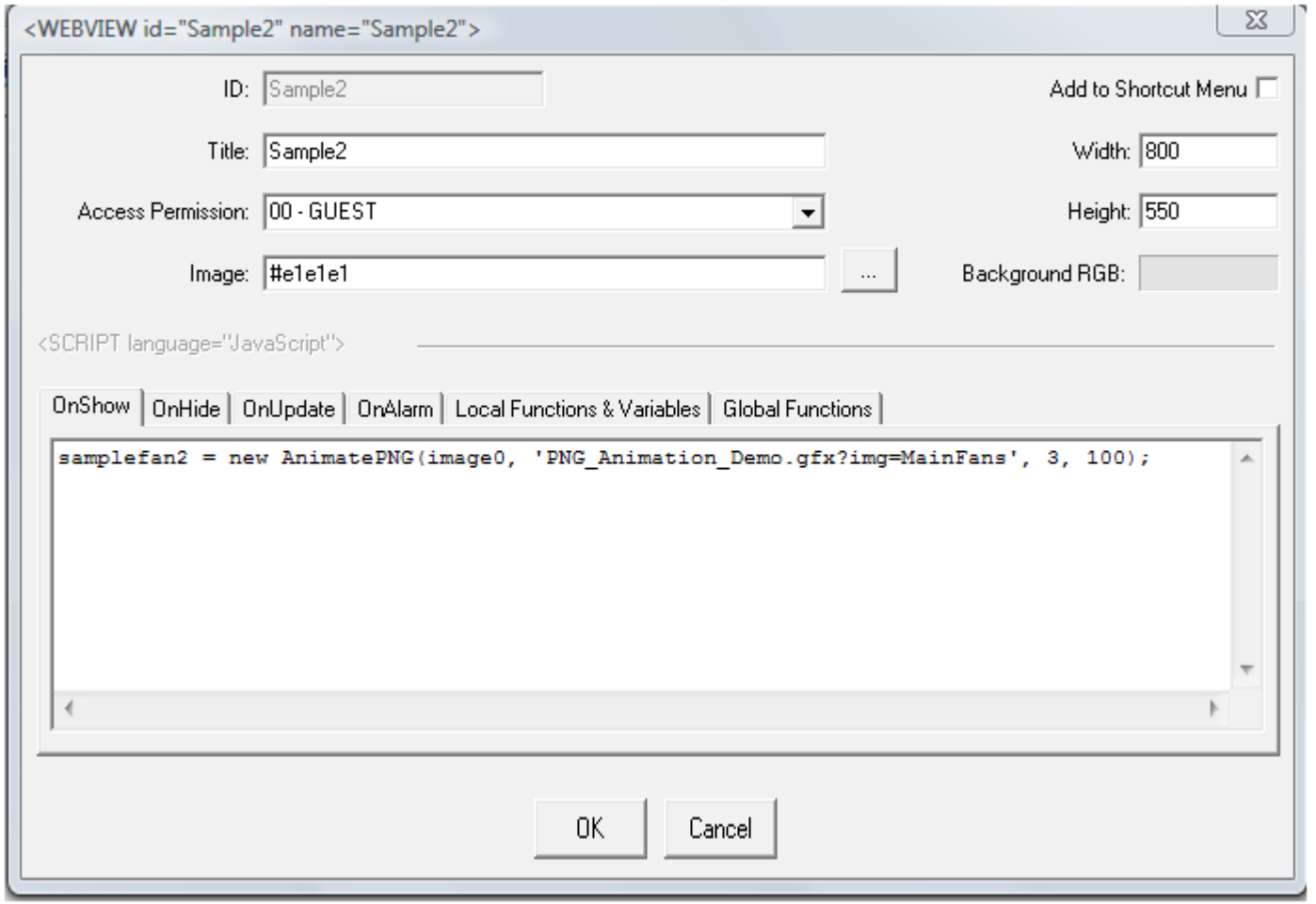


Hit Control-V to paste the text from the clipboard. This text provides the path to the images that WebLink will cycle through to create an animation effect



Next, enter 3 as the third argument to the AnimatePNG function because this is the number of images we noted down earlier

To create the animation effect WebLink will briefly show MainFans_0.png, then show MainFans_1.png, then MainFans_2.png, then cycle back to MainFans_0.png

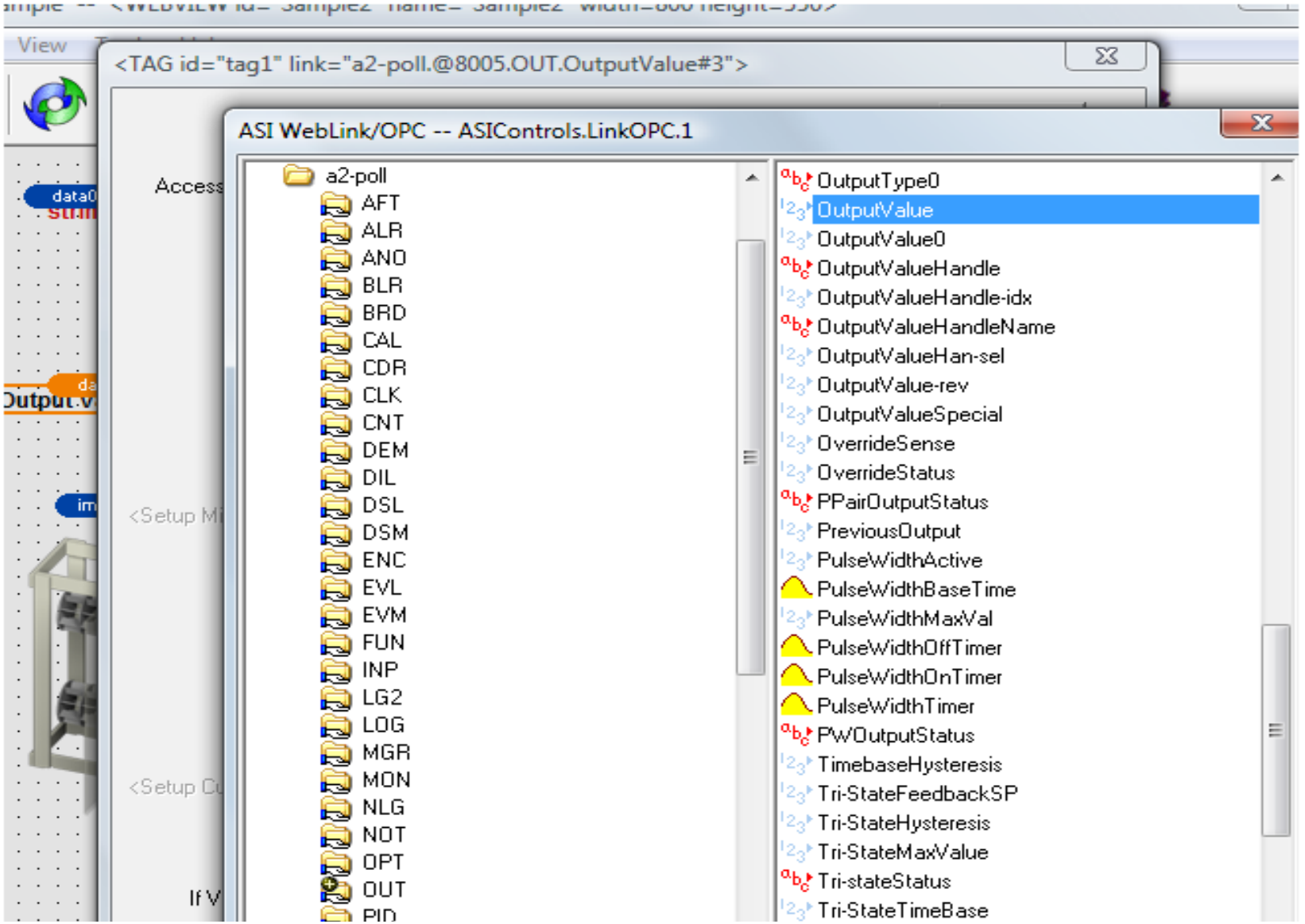


The fourth and last argument to the AnimatePNG function tells WebLink the number of milliseconds to display each image in the series before cycling to the next image, we use 100

Complete line:

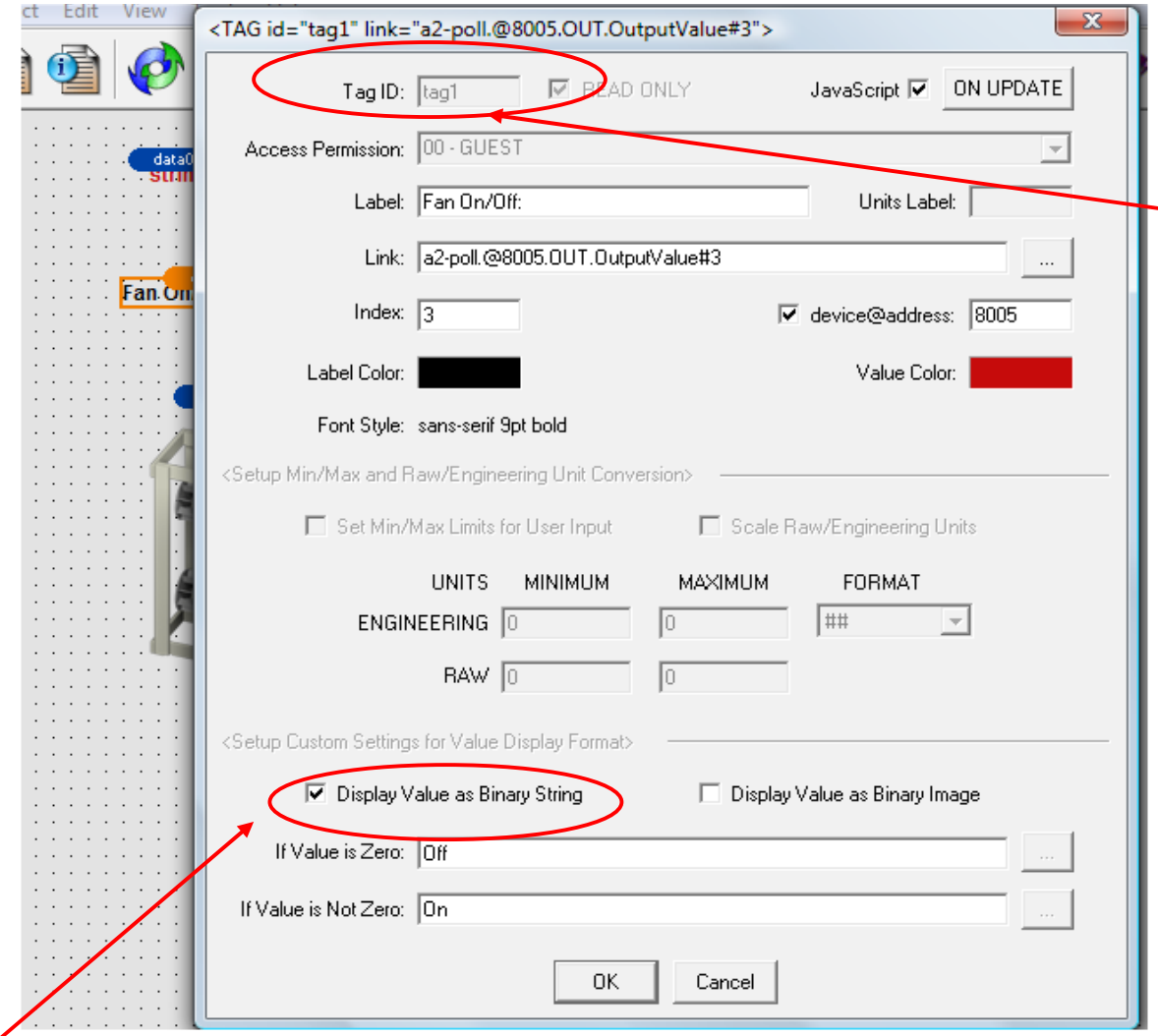
```
samplefan2 = new AnimatePNG(image0, 'PNG_Animation_Demo.gfx?img=MainFans',3,100);
```

Click OK to save changes



Open the OPC tag browser. Choose a spare Binary Output (OUT-3) that you can override on and off to watch the animation go on/off.

Drag OutputValue onto the screen, the new “tag1” element is initially labeled “Output Value:”



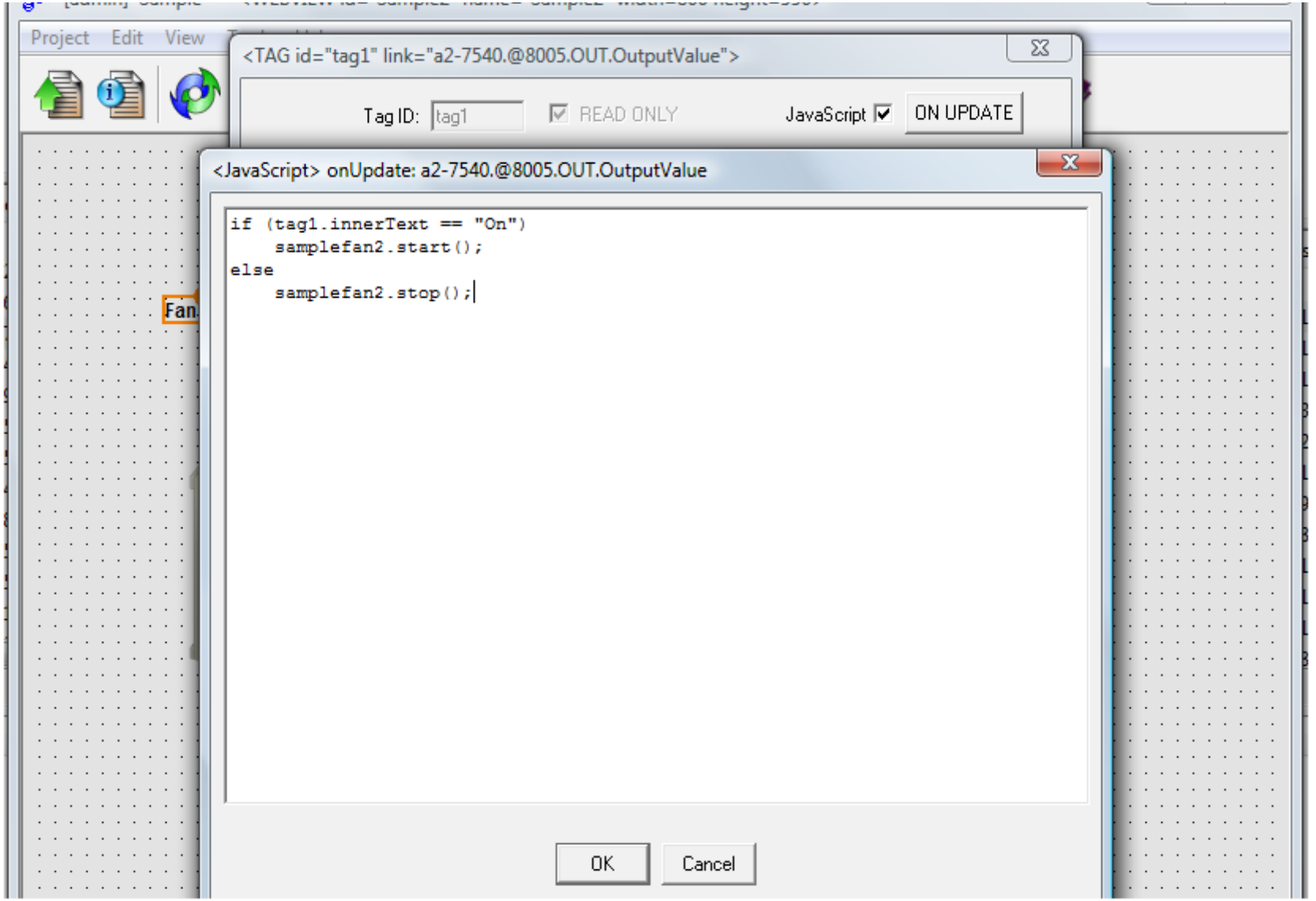
Note that the ID for the element we just dragged onto the screen is “tag1”

We will need this in our javascript

Choose “Display Value as Binary String”, with “Off” for zero and “On” for non-zero values

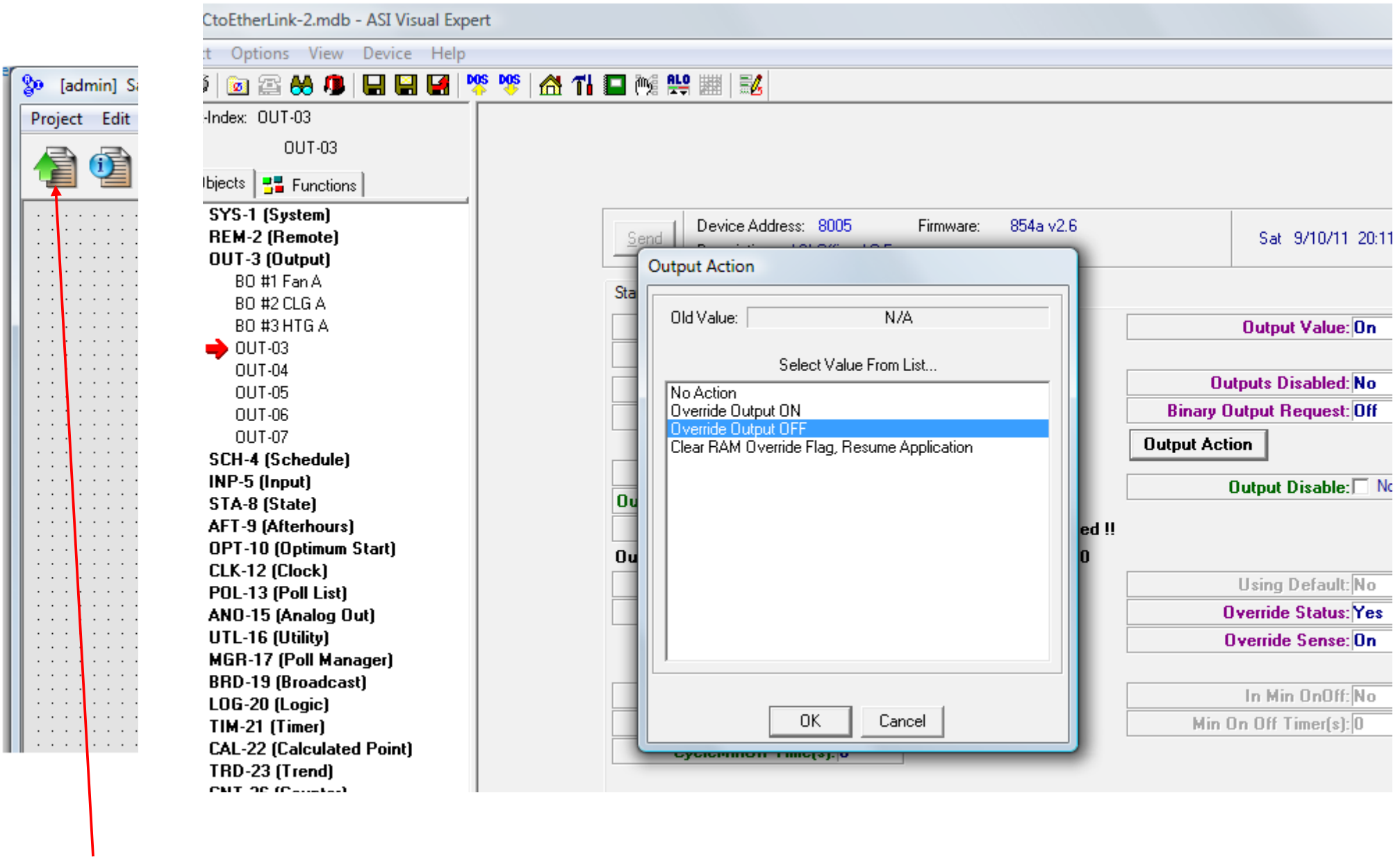
Enter a label “Fan On/Off:”

Click the ON UPDATE button in the top right corner so we can add some javascript that will be executed each time the output value changes



Enter the following javascript, then click OK to save changes

```
if (tag1.innerText == "On")
    samplefan2.start();
else
    samplefan2.stop();
```



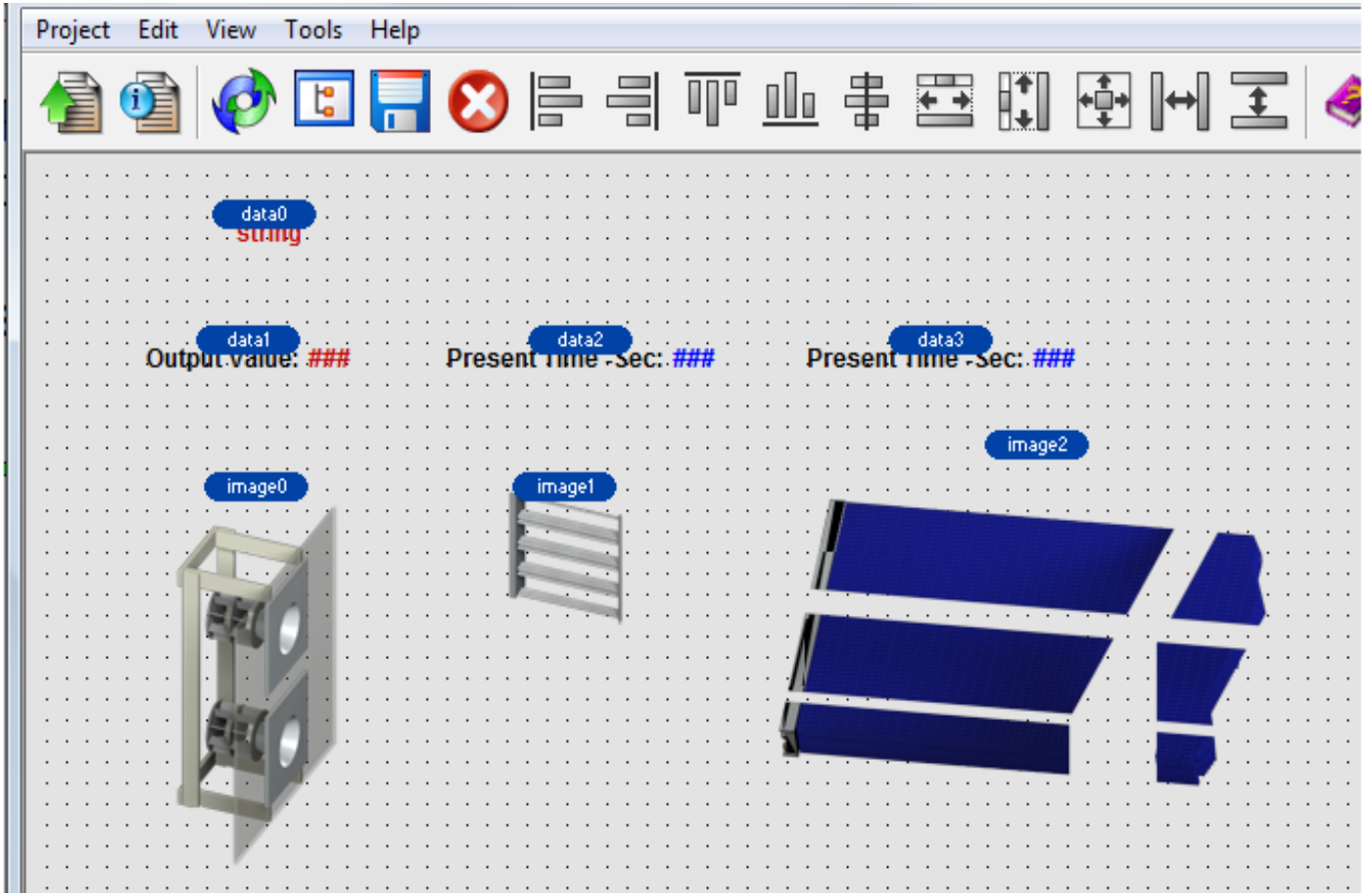
Click the green arrow in WebLink to start a web session.

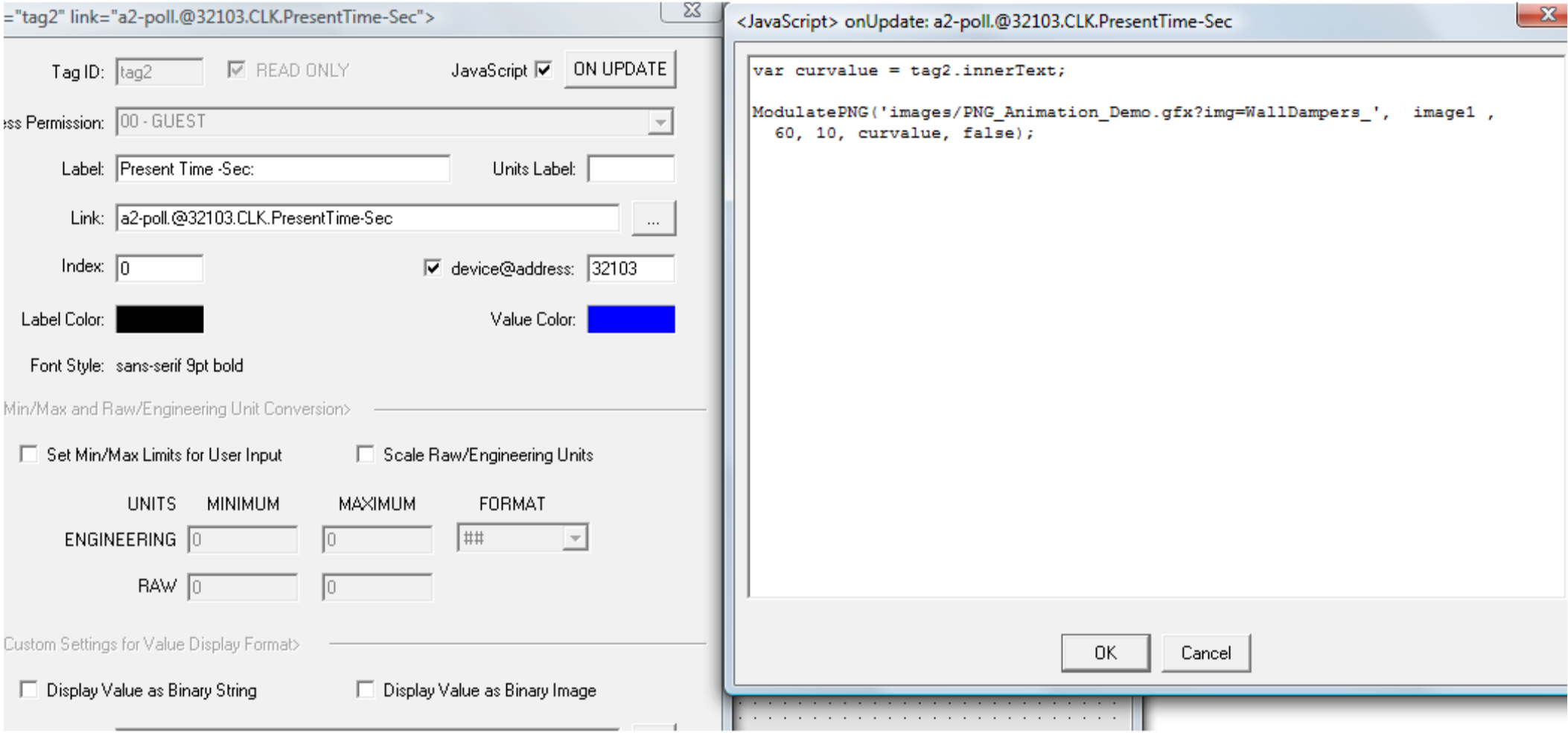
You should see the fan start and stop as you override the output on and off in Expert

Add a second image to the screen to do Damper modulation, this will be “image1”. Select WallDampers_0.png image from the GFX library as the initial image

Use the OPC tag browser to pull “tag2” on screen with the CLK.PresentTime-Sec value to display seconds ticking over, place the tag above the damper image

Add an image (“image2”) for Coil modulation, use the CoolCoil_ series from the PNG library. Add “tag3” to display CLK.PresentTime-Sec value and place the tag above the cooling coil





Open data2 properties, click the ON UPDATE button, copy/paste the following script:

```
var curvalue = tag2.innerText;

ModulatePNG('images/PNG_Animation_Demo.gfx?img=WallDampers_',
  image1 ,60, 10, curvalue, false);
```

Click OK to save changes

Note the **ModulatePNG** function is passed 6 arguments, in order these are:
- the root path for the images
- the element (image1, image2, etc.) that holds the image being modulated
- the maximum value that will be read (minimum is always zero)
- the number of images to use in the modulation (X images, numbered 0 through X-1)
- the current value, which must be within the range of 0 to maximum
- whether to reverse the order of modulation, a true/false parameter

Remember that Javascript is case-sensitive, so ModulatePNG is not the same as modulatepng (and using the latter spelling would generate an error in our example)

In javascript it is usually best to use single quotes to encapsulate string values rather than double-quotes, especially if you are copying/pasting from/to an editor other than WebLink

You can wrap lines of script by using the Enter or Return key and then using spaces to align the text (if desired)

Do not use tabs to align code in WebLink, use only spaces

```
function ModulatePNG(rootImagePath, imageobj, maximum, image_count, curvalue, reverse) {

    curvalue = parseFloat(curvalue);   //strip any non-numeric characters such as Units Label

    var step_size = maximum / image_count;       // calculate percent of total value for each image

    if (reverse) {
        for (i = 0; i < image_count; i++) {     // loop UP from something_0.png to something_X.png
            if (curvalue < step_size * i) {
                imageobj.src = rootImagePath + i + ".png";
                break;
            }
        }
    } else {
        for (i = 0; i < image_count; i++) {     // loop DOWN from something_0.png to something_X.png

            if (curvalue > step_size * (image_count - i - 1)) {
                imageobj.src = rootImagePath + i + ".png";
                break;
            }
        }
    }
}
```

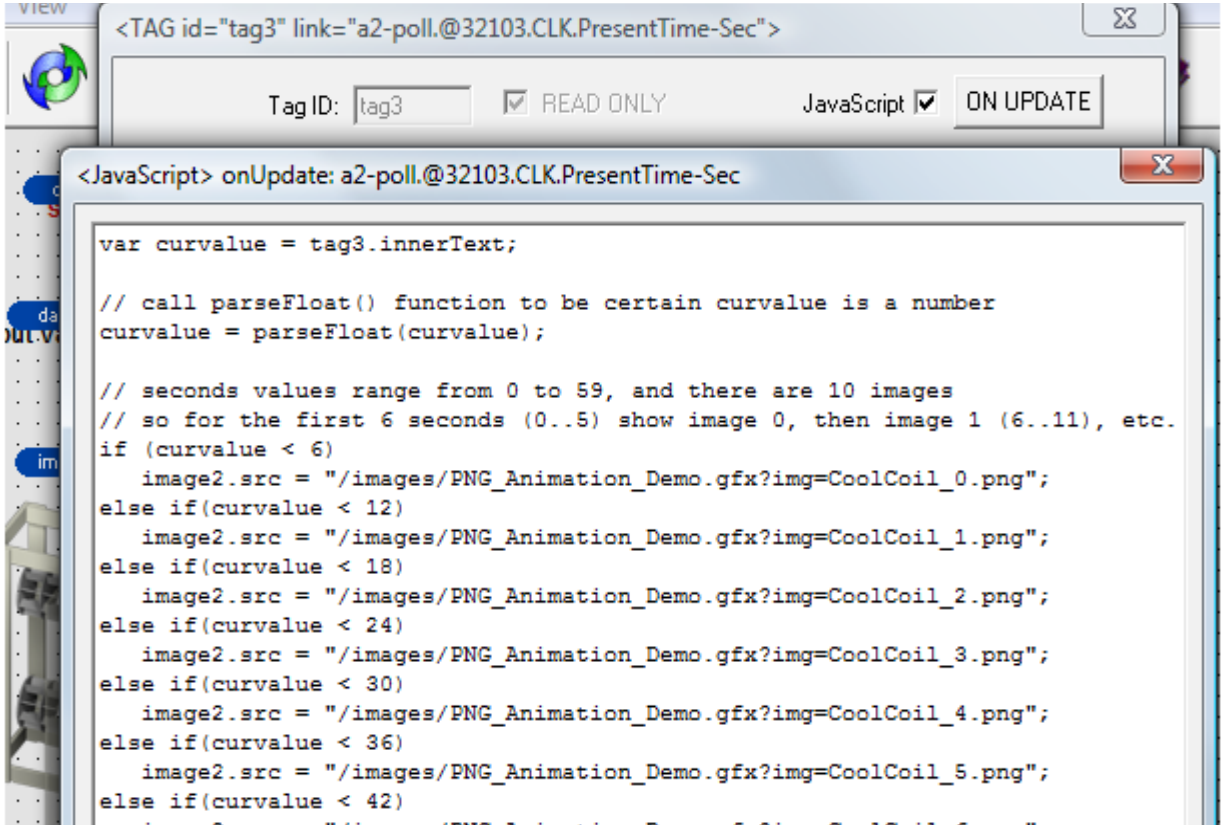
The ModulatePNG() function works by dividing the maximum value into N equal **intervals**, where N is the number of images available for modulation. If the current value is in the first interval show the first image, if it's in the second interval show the second image, if current value in the 3rd interval show the 3rd image, etc.

Call parseFloat() because if "curvalue" variable were non-numeric then comparisons to step_size would be done alphabetically rather than numerically, which could break our modulation logic

*The "step_size" variable is the **interval**, it is determined by dividing maximum value by number of images*

Determine which interval our current value ("curvalue") is in, then assign the image

A "break" statement tells the script to break out of the loop because we are done



Open data3 properties, click the ON UPDATE button, this time we use a ladder of “if-then-else” statements instead of calling the ModulatePNG() function

*If value is less than maximum value divided by the number of images (6 = 60/10) show image #0, ELSE if value is less than **2** * max / # images show image #**1**, ELSE if value is less than **3** * max / # images show image #**2**, ELSE if value is less than **4** * max / # images show image #**3**, etc.*

The logic in this script is perhaps easier to understand than code in the ModulatePNG() function, but a tradeoff is there are more opportunities to make mistakes when copying and pasting this block of code to use on another page, or when reversing the direction

Click OK to save changes

Click the green arrow to start a web session and let a few seconds go by

You should see the dampers modulating every 6 seconds, the sequence is:

* closed from 0 to 5 seconds,
* 10% open from 6 to 11 seconds,
* 20% open from 12 to 17 seconds,
* 30% open from 18 to 23 seconds, etc. up to
* 100% open from 54 to 59 seconds

The cooling coil should glow in this sequence:

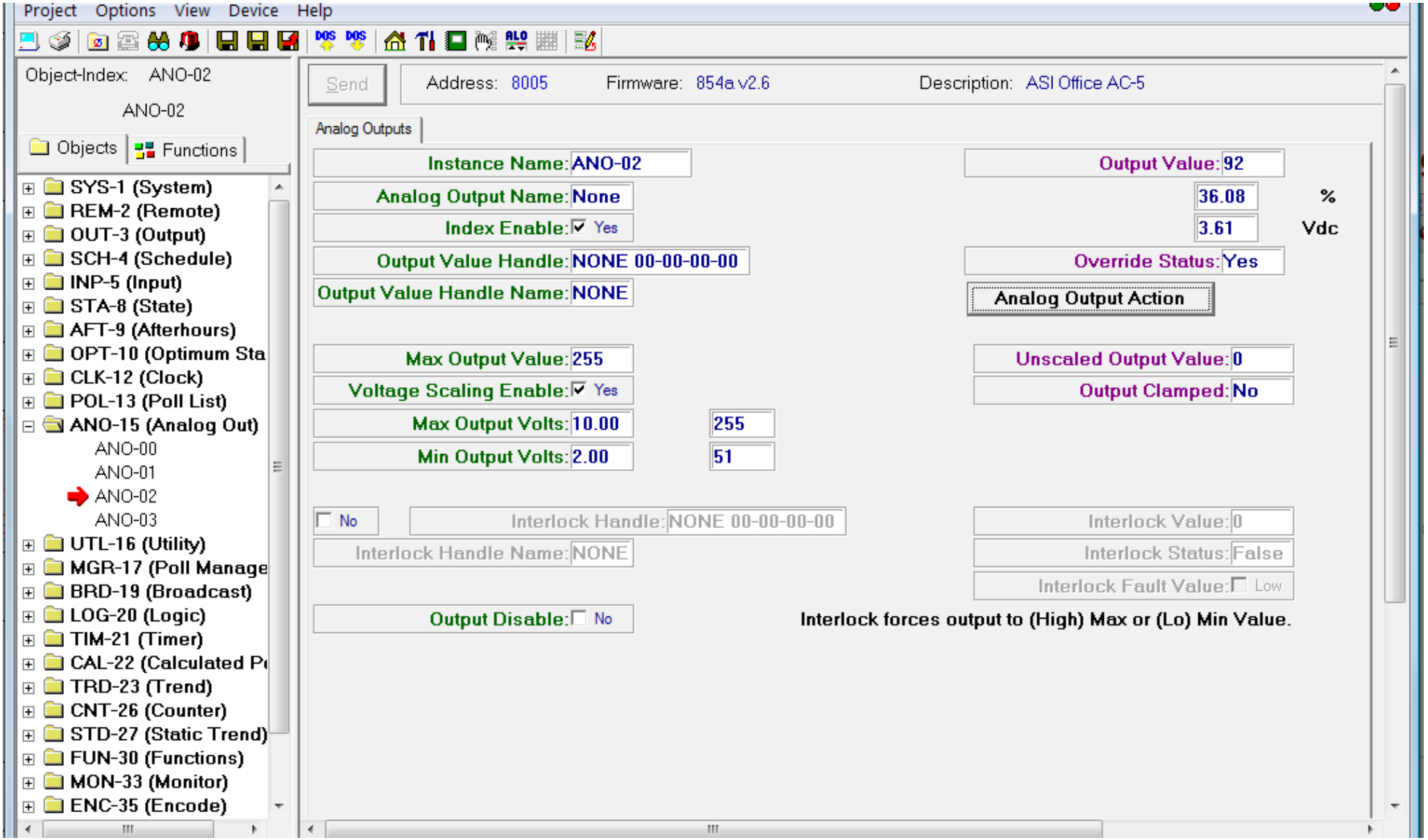
* dark from 0 to 5 seconds,
* 10% glow from 6 to 11 seconds,
* 20% glow from 12 to 17 seconds,
* 30% glow from 18 to 23 seconds, etc.
...
* 100% glow from 54 to 59 seconds

TIP: the last argument to the ModulatePNG function is a true/false value to reverse the order in which PNG graphics are modulated.

Modulating in forward direction (value=false) iterates from something_0.png to something_X.png with values ranging from 0 to 100%, while modulating in reverse (value=true) goes from something_X.png to something_0.png with 0 to 100% values

Next is an example of modulating images for a Hot Coil and Valve

First, setup Analog Output to provide a 2 to 10 volt signal (values from 51 to 255)



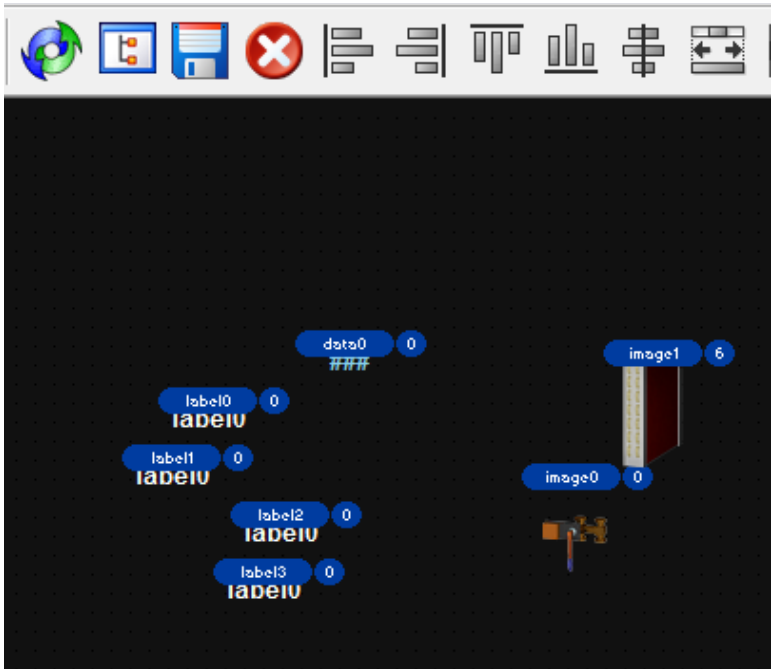
WebLink Elements for HTG Coil & Valve Example

tag0 – OPC data tag mapped to analog output

label0 – shows image # for HTG coil (0 to 9)
label1 – shows image number for Valve

label2 – display voltage
label3 – display % (0 to 100)

img0 – holds the valve
img1 – holds the coil



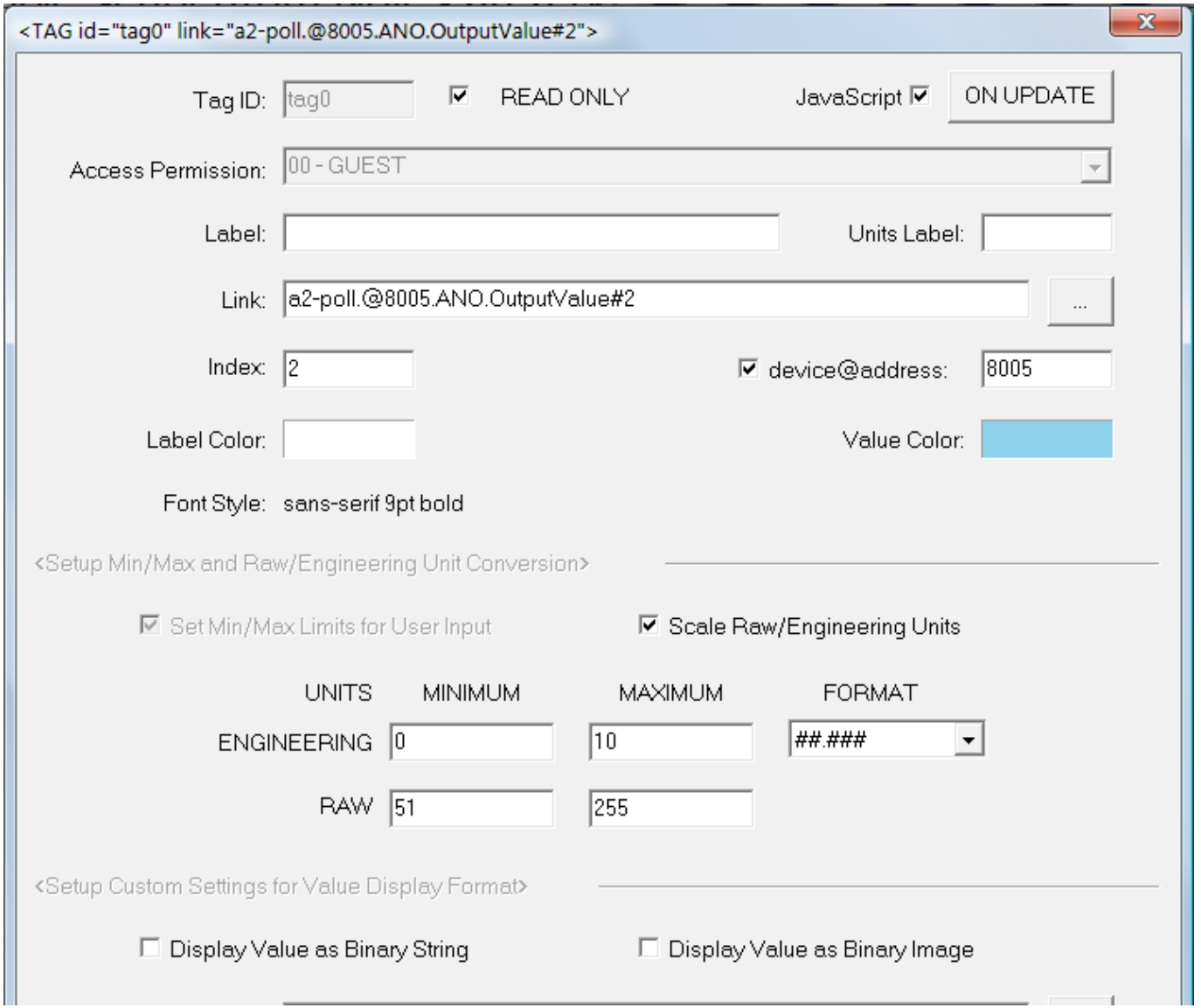
We use labels to show the voltage and the % as an example of re-using a single data tag rather than adding more tags to show voltage and %. Want to minimize data traffic between the OPC server and controller for best performance

You can download the example WebLink .WEB screen here:

www.asicontrols.com/var/training/how_to_animate_png/Hot_Coil_and_Valve.web

Add an OPC data tag (“tag0”) mapped to the OutputValue of the Analog Output

Scale the RAW input (51 to 255) to 0 to 10 because we will modulate using 10 images. Format is ##.### to provide 3 significant digits



Copy the sample script (from Notes area below, or from Text version of this page) to the clipboard and Paste it into the ON UPDATE event of the new OPC data tag

First get the value from the tag

Javascript parseFloat() function removes any non-numeric characters

Limit value range to 0 to 10

Truncate any fractions, convert 10 to 9

Assign the valve image

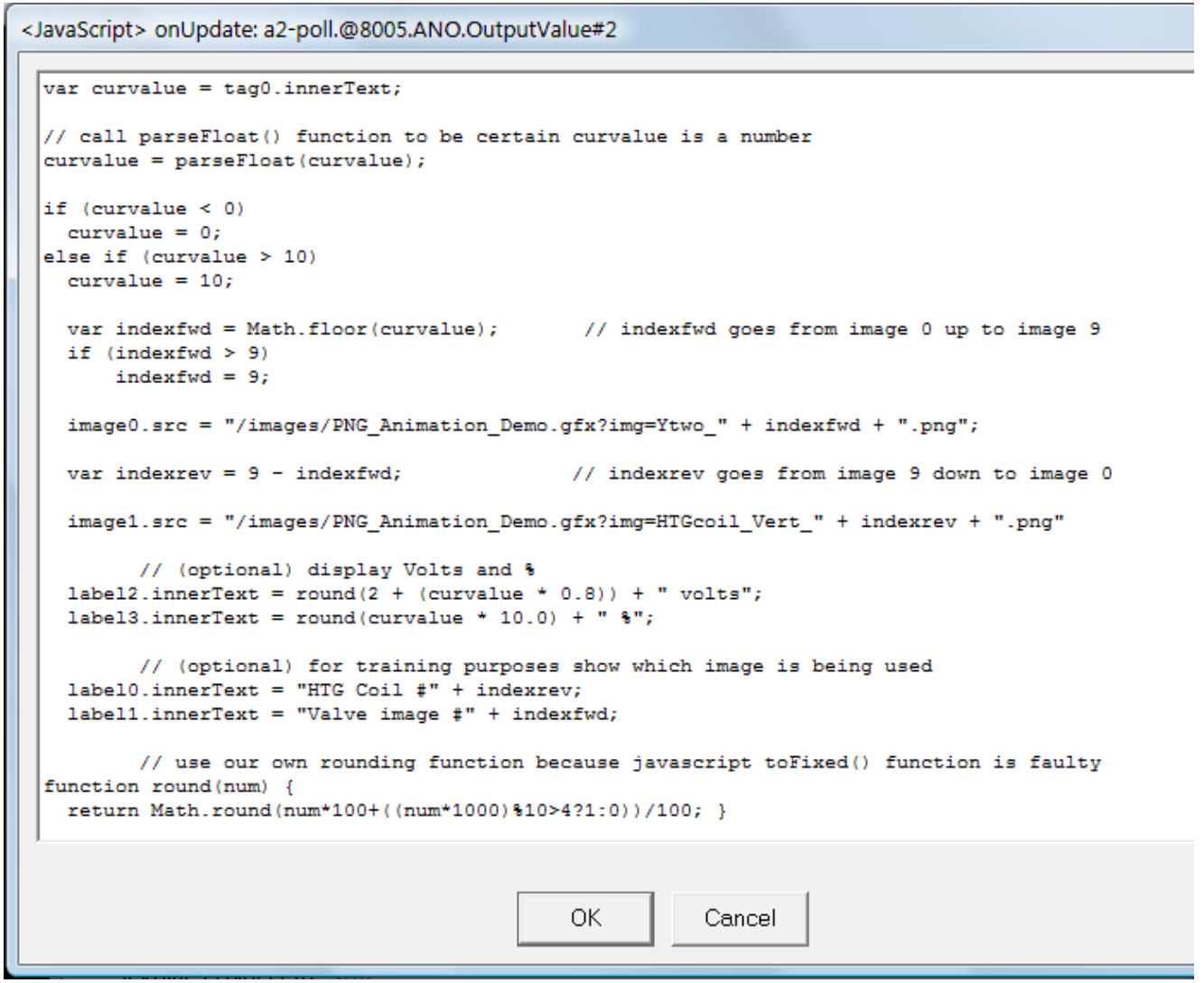
Calculate the reverse image index

Assign the heating coil image

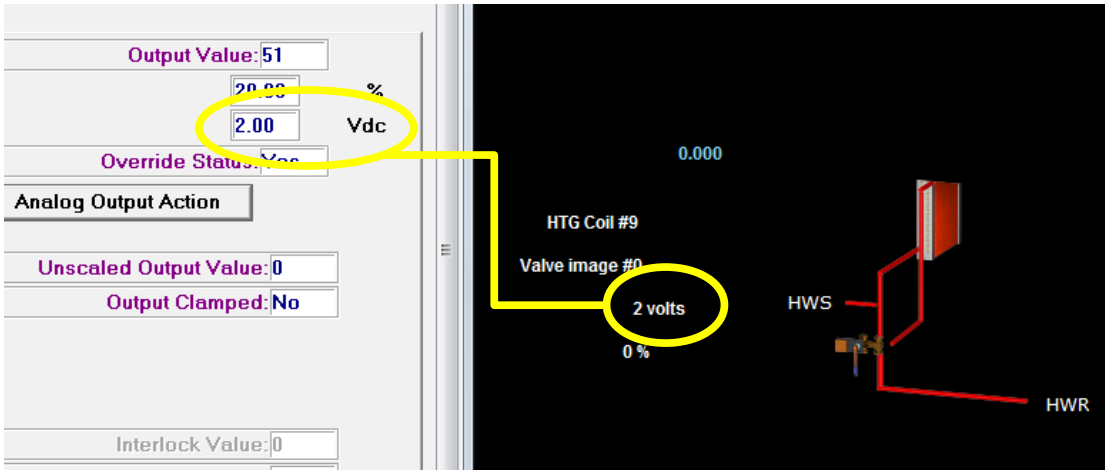
Display voltage and %

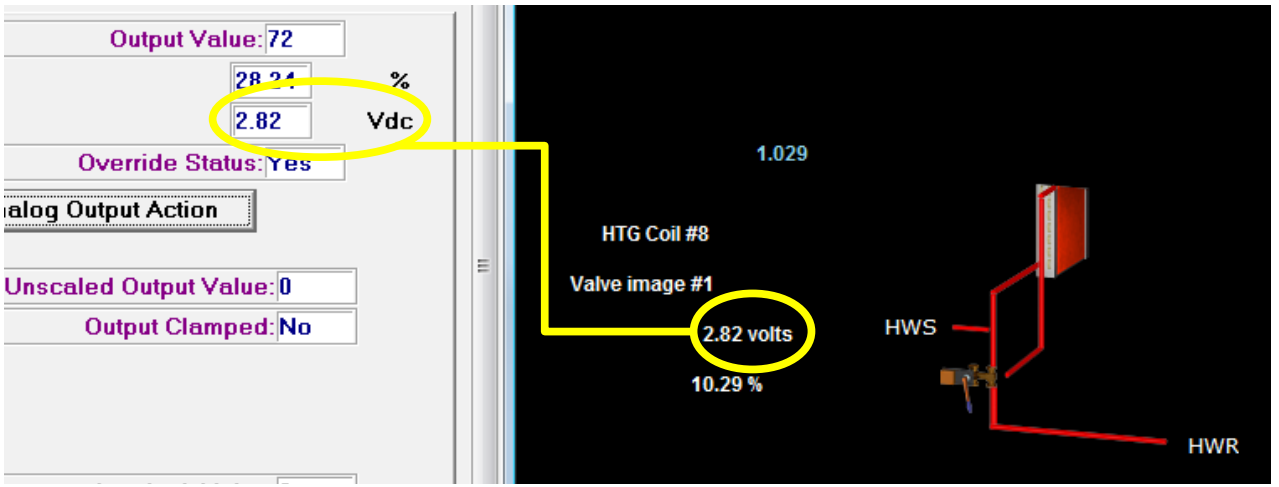
Display image numbers for clarity

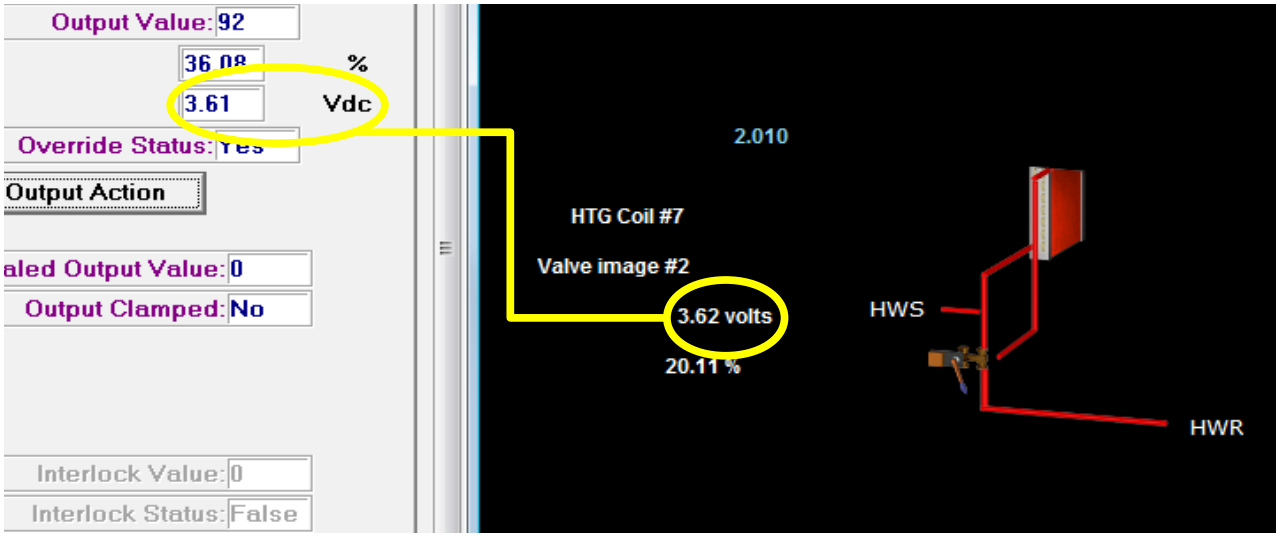
Javascript toFixed() function does rounding, but it does not do a good job, so we use our own rounding function



Some screen shots taken while validating this example







You may see some very slight differences in the voltage signal that are caused by number rounding issues in javascript

Our script prevents any values exceeding 100% or below 0%, even if the values were somehow out of range